NASA Conference Publication 3251

Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94)

Volume I

(March 21-22)

Proceedings of a conference sponsored by The American Institute of Aeronautics & Astronautics and the National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas

March 21-24, 1994

The American Institute of Aeronautics and Astronautics

National Aeronautics and Space Administration

NASA Conference Publication 3251

Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94)

Volume I
(March 21-22)

Jon D. Erickson, Editor
NASA Lyndon B. Johnson Space Center
Houston, Texas

Proceedings of a conference sponsored by
The American Institute of Aeronautics & Astronautics
and the National Aeronautics and Space Administration,
Lyndon B. Johnson Space Center, Houston, Texas
March 21-24, 1994

In cooperation with
American Association of Artificial Intelligence,
IEEE Robotics and Automation Society,
National Service Robot Association, Robotic Industries Association,
Society of Optical Instrumentation Engineers,
AIAA Space Automation and Robotics Technical Committee,
Clear Lake Council of Technical Societies, AIAA Houston Section,
ISA Clear Lake Section, and IEEE Galveston Bay Section

National Aeronautics and
Space Administration

Preface

The formation of the AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94) was originally proposed because of the strong belief that America's problems of global economic competitiveness and job creation and preservation can partly be solved by the use of intelligent robotics, which are also required for human space exploration missions. It was also recognized that in the applications-driven approach there are a far greater set of common problems and solution approaches in field, factory, service, and space applications to be leveraged for time and cost savings than the obvious differences in implementation details would lead one to believe. This insight, coupled with a sense of national urgency, made a continuing series of conferences to share the details of the common problems and solutions across these different fields of application not only a natural step, but a necessary one. Further, it was recognized that a strong focusing effort is needed to move from recent factory-based robot technology into robotic systems with sufficient intelligence, reliability, safety, multi-task flexibility, and human/machine interoperability to meet the rigorous demands of each of these fields of application. The scope of this effort is beyond the capability of the private sector alone, government alone, or academia alone. Cooperation by all interested parties is essential to achieve the needed investments and maximize the benefits from innovation.

The first AIAA/NASA conference on intelligent robotics is a clear success, judging from the quality and number of papers for presentation and manuscripts collected in these proceedings. Also, having the proceedings available at the conference is important to communication effectiveness and efficiency; the authors are to be congratulated for meeting the deadline. Having Dr. Joseph Engelberger, Chief Executive Officer of Transitions Research Corporation, present the keynote address emphasizing the applications-driven approach to technology development sets the correct tone and background for getting on with the job of strategic investment in and development of intelligent robotics through cooperative national efforts.

The papers in these proceedings are evidence that users in each field, manufacturers and integrators, and technology developers are rapidly increasing their understanding of the "whats" and "hows" of integrating robotic systems on Earth and in space to accomplish economically important tasks requiring mobility and manipulation. The 21 sessions of technical papers in seven tracks plus two plenary sessions cover just the tip of this major progress, but reveal its presence nonetheless.

The contents pages of these proceedings do not necessarily reflect the final program nor the arrangement of presentations in sessions. The conference brochure provides the information.

Appreciation goes to the Steering Committee members, Program Committee members, Track chairs, and Session chairs who are all so essential to making this a successful conference through the voluntary giving of their time and efforts. Special thanks and personal admiration go to Larry Seidman, Zafar Taqvi, Hatem Nasr, Mary Stewart, Donna Maloy, and Dottye Hamblin for their efforts to make this conference happen.

Jon D. Erickson

Conference Chair
Jon D. Erickson

Contents

Volume I
(March 21-22)

First Plenary Session: *Jon D. Erickson, Conference Chair*

Intelligent Robotics Can Boost America's Economic Growth
Jon D. Erickson, *Chief Scientist for Automation and Robotics Division, NASA Johnson Space Center, Houston, Texas* .... 1 -1

Field Track: *Harvey B. Meieran, Chair*

Nuclear Industry Session: Jack J. Judge, Jr., Chair

Teleoperated Systems for Nuclear Reactors Inspection and Maintenance
V. P. Dorokhov, *TECHNIKA, St. Petersburg, Russia*; D. V. Dorokhov, *Baltic State Technical University, St. Petersburg, Russia*; A. P. Eperin, *Leningrad Atomic Power Station, Sosnoviy Bor, Russia* 8-2

ARK: Autonomous Mobile Robot in an Industrial Environment
S. B. Nickerson, J. R. R. Service, and D. Wilkes, *Ontario Hydro Technologies, Toronto, Canada*; P. Jasiobedzki, A. Jepson, B. Down, D. Terzopoulos, and J. Tsotsos, *University of Toronto, Toronto, Canada*; M. Jenkin and E. Milios, *York University, North York, Canada*; N. Bains and T. Campbell, *Atomic Energy of Canada Ltd., Mississauga, Canada* .... 12 - 3

Biologically-Inspired Hexapod Robot Design and Simulation
Kenneth S. Espenschied and Roger D. Quinn, *Case Western Reserve University, Cleveland, Ohio* 21- 4

Odysseus Autonomous Walking Robot: The Leg/Arm Design
N. G. Bourbakis, M. Maas, A. Tascillo, and C. Vandewinckel, *Binghamton University, Binghamton, New York* .... 29 - 5

Factory Track: *Michael B. Leahy, Jr., Chair*

Agile Manufacturing Session: Joseph M. Loibl, Chair

A Vision Advisor System for Flexible Manufacturing
Jim Eilbert, William Lim , Jay Mendelsohn, Ron Braun and Mike Yearwood, *Grumman Corporation, Bethpage, New York* .... 37 -6

Application of the Modular Automated Reconfigurable Assembly System Concept to Adaptable Vision Gauging and Parts Feeding
Andre By and Ken Caron, *Tufts University, Medford, Massachusetts*; Michael Rothenberg and Vic Sales, *Productivity Technologies, Inc., Sunnyvale, California* .... 47 - 7

Agile Manufacturing and The Factory of the Future
Joseph M. Loibl and Terry Bossieux, *Ford Motor Co. - AMTAC, Dearborn, Michigan* .... 56 - 8

Contents
(continued)

Confessions of a Robot Lobotomist
R. Marc Gottshall, *Boeing Company, Commercial Airplane Group, Seattle, Washington* 63

Integration of Vision and Robotic Workcell
Terry Bossieux, *Ford Motor Co. - AMTAC, Dearborn, Michigan* ..... 73 -10

Service Track: *John M. Holland, Chair*

Security/Building Monitoring Session: *Celeste DeCorte, Chair*

Meeting the Challenges of Installing a Mobile Robotic System
Celeste DeCorte, *Cyberomotion, Inc., Salem, Virginia* 78-11

Design of an Autonomous Exterior Security Robot
Scott D. Myers, *Robotic Systems Technology, Westminster, Maryland* 82-12

Task Automation in a Successful Industrial Telerobot
Sammy L. Jones, *Remotec, Inc., Oak Ridge, Tennessee*; Phillip F. Spelt, *Oak Ridge National Laboratory, Oak Ridge, Tennessee* ..... 88 -13

Controlling Multiple Security Robots in a Warehouse Environment
H. R. Everett, G. A. Gilbreath, T. A. Heath-Pastore, and R. T. Laird, *Naval Command Control & Ocean Surveillance Center, San Diego, California* ..... 93 -14

Space Track: *Joseph C. Parrish, Chair*

On-Orbit Applications I Session: *Joseph C. Parrish, Chair*

Technology Transfer and Evaluation for Space Station Telerobotics
Charles R. Price and Lebarian Stokes, *NASA Johnson Space Center, Houston, Texas*; Myron A. Diftler, *Lockheed Engineering & Sciences Company, Houston, Texas* 103-15

Space Flight Manipulator Technologies and Requirements as Developed for the NASA Flight Telerobotic Servicer (FTS)
John T. Chladek, *NASA Johnson Space Center, Houston, Texas*; William M. Craver, *Lockheed Engineering & Sciences Company, Houston, Texas* ..... 112 -16

A Space Station Robot Walker and Its Shared Control Software
Yangsheng Xu and Ben Brown, *Carnegie Mellon University, Pittsburgh, Pennsylvania*; Shigeru Aoki and Tetsuji Yoshida, *SHIMIZU Corporation, Tokyo, Japan* 123

Technology for Robotic Surface Inspection in Space
Richard Volpe and J. Balaram, *NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California* 131

A Highly Redundant Robot System for Inspection
Thomas Lee, Tim Ohms, and Samad Hayati, *NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California* 142

Free-Floating Dual-Arm Robots for Space Assembly
Sunil K. Agrawal and M. Y. Chen, *Ohio University, Athens, Ohio* 149

Contents
(continued)

Robotic Sensing, Vision, and Perception Track: *Hatem Nasr, Chair*

Vision & Sensing Technologies for Autonomous Robots Session: *Bir Bhanu, Chair*

Design and Control of Active Vision Based Mechanisms for Intelligent Robots
Liwei Wu and Michael M. Marefat, *University of Arizona, Tucson, Arizona* .... 158 -21

Vision Based Object Pose Estimation for Mobile Robots
Annie Wu, Clint R. Bidlack, Terry Weymouth, Arun Katkere, and Roy Feague,
The University of Michigan, Ann Arbor, Michigan .... 168 -22

Unsupervised Texture Image Segmentation by Improved Neural Network ART2
Zhiling Wang, G. Sylos Labini, and R. Mugnuolo, *Italian Space Agency, Matera, Italy*; Marco De Sario, *University of Bari, Bari, Italy* .... 175 -23

Microwave Vision for Robots
Leon Lewandowski and Keith Struckman, *Lockheed Sanders, Inc., Nashua, New Hampshire* .... 181 -24

An Electromagnetic NonContacting Sensor for Thickness Measurement in a Dispersive Media
Robert L. Chufo, *U.S. Dept. of the Interior, Bureau of Mines, Pittsburgh, Pennsylvania* .... 190 -25

Perception System and Functions for Autonomous Navigation in a Natural Environment
Raja Chatila, Michel Devy, Simon Lacroix, and Matthieu Herrb, *Centre National de la Recherche Scientifique, Laboratoire d'Automatique et d'Analyse des Systems, Toulouse, France* .... 195 -26

Planning, Reasoning, and Control Track: *R. Peter Bonasso, Chair*

Situated Control and Low-Level Control Session: *William Lim, Chair*

Fuzzy Logic Based Robotic Controller
Farouk G. Attia, M. Upadhyaya, and S. Ramchandani, *University of Houston, Houston, Texas* .... 206 -27

Vehicle Following Controller Design for Autonomous Intelligent Vehicles
C. C. Chien, R. Mayr, and M. C. Lai, *University of Southern California, Los Angeles, California* .... 212 [illegible]

The Real-World Navigator
Illah R. Nourbakhsh, Craig Becker, Marko Balabanovic, and Sarah K. Morse, *Stanford University, Stanford, California* .... 223 [illegible]

A Streamlined Software Environment for Situated Skills
Sophia T. Yu, Marc G. Slack, and David P. Miller, *The MITRE Corporation, McLean, Virginia* .... 233 -30

Situational Reaction and Planning
John Yen and Nathan J. Pfluger, *Texas A&M University, College Station, Texas* .... 240 -31

Contents
(continued)

Autonomous Mobile Robot Teams
Arvin Agah and George A. Bekey, *University of Southern California, Los Angeles, California* 246-32

Systems Technology and Architectures Track: *Arthur C. Sanderson, Chair*

Robotic Systems Architectures Session: *Ian D. Walker, Chair*

Building Brains for Bodies
Rodney Allen Brooks and Lynn Andrea Stein, *Massachusetts Institute of Technology, Cambridge, Massachusetts* 253-33

Object-Based Task-Level Control: A Hierarchical Control Architecture for Remote Operation of Space Robots
Homer D. Stevens, Eric S. Miles, Stephen M. Rock, and Robert H. Cannon, *Stanford University, Stanford, California* 264-34

Task-Level Control for Autonomous Robots
Reid Simmons, *Carnegie Mellon University, Pittsburgh, Pennsylvania* 275-35

A Survey of NASA and Military Standards on Fault Tolerance and Reliability Applied to Robotics
Joseph R. Cavallaro and Ian D. Walker, *Rice University, Houston, Texas* 282-36

A Performance Analysis Method for Distributed Real-Time Robotic Systems: A Case Study of Remote Teleoperation
Donald R. Lefebvre and Arthur C. Sanderson, *Rensselaer Polytechnic Institute, Troy, New York* 287-37

Predictive Sufficiency and the Use of Stored Internal State
David J. Musliner, *The University of Maryland, College Park, Maryland*; Edmund H. Durfee, and Kang G. Shin, *The University of Michigan, Ann Arbor, Michigan* 298-38

Using Generic Tool Kits to Build Intelligent Systems
David J. Miller, *Sandia National Laboratories, Albuquerque, New Mexico* 306-39

Field Track: *Harvey B. Meieran, Chair*

Environmental Restoration, Waste Management, and Hazardous Operation Session: *Harvey B. Meieran, Chair*

A Reactive System for Open Terrain Navigation: Performance and Limitations
Dirk Langer, Martial Hebert, and J. K. Rosenblatt, *Carnegie Mellon University, The Robotics Institute, Pittsburgh, Pennsylvania* 315-40

The Road Plan Model - Information Model for Planning Road Building Activities
Rafaela K. Azinhal, *UNINOVA, Monte da Caparica, Portugal*; Fernando Moura-Pires, *Universidade Nova de Lisboa, Monte da Caparica, Portugal* 322-41

Design Reuse Experience of Space and Hazardous Operations Robots
P. Graham O'Neil, *Lockheed Engineering & Sciences Company, Houston, Texas* 333

Contents
(continued)

A Multi-Mode Manipulator Display System for Controlling Remote Robotics Systems
Michael J. Massimino, M. F. Meschler, and A. A. Rodriguez, *McDonnell Douglas Aerospace, Houston, Texas* .... 339 - 43

Factory Track: *Michael B. Leahy, Jr., Chair*

Robotic Remanufacturing Session: *Michael B. Leahy, Jr., Chair*

Programmable Automated Welding System
Martin D. Kline, *Lynchburg, Virginia*; Thomas E. Doyle, *Alliance, Ohio* .... 348 - 44

Robotic NDE Inspection of Advanced Solid Rocket Motor Casings
Glenn E. McNeelege and Chris Sarantos, *Babcock & Wilcox, Lynchburg, Virginia* .... 354 - 45

Automation For Nondestructive Inspection of Aircraft
M. W. Siegel, *Carnegie Mellon University, Pittsburgh, Pennsylvania* .... 367 - 46

Graphical Simulation for Aerospace Manufacturing
Brian K. Christensen and Majid Babai, *NASA Marshall Space Flight Center, Huntsville, Alabama*; Christopher Bien, *Deneb Robotics, Inc. Auburn Hills, Michigan* .... 378 - 47

The Automated Aircraft Rework System (AARS) -- A System Integration Approach
Michael J. Benoit, *Mercer University, Warner Robins, Georgia* .... 381 - 48

Automated Inspection of Turbine Blades: Challenges and Opportunities
Manish Mehta, Joseph C. Marron, Robert E. Sampson, and George M. Peace, *ERIM, Ann Arbor, Michigan* .... 393 - 49

Service Track: *John M. Holland, Chair*

Healthcare Session: *W. Stuart Lob, Chair*

TRC Research Products: Components for Service Robots
W. Stuart Lob, *Transitions Research Corporation, Danbury, Connecticut* .... 399 - 50

An update on "Lab Rover": a Hospital Material Transporter
Paul Mattaboni, *Cyberotics, Inc., Waltham, Massachusetts* .... 405 - 51

A Robotic Wheelchair
David P. Miller, *The KISS Institute, Reston, Virginia*; Edward Grant, *Power Concepts Incorporated, Fresno, California* .... 407 - 52

Dexterity Enhancement in Microsurgery Using Telemicro-Robotics
Steve Charles andSteve Charles, M. D., *Micro-Dexterity Systems, Inc., Memphis, Tennessee* .... 412 - 53

An Intelligent Robotic Aid System for Human Services
Kazuhiko Kawamura, Sugato Bagchi, and Todd Park, *Vanderbilt University, Nashville, Tennessee* .... 413 - 54

Contents
(continued)

Space Track: *Joseph C. Parrish, Chair*
On-Orbit Applications II Session: *David B. Lavery, Chair*

An Intelligent Robot for Helping Astronauts
Jon D. Erickson, Keith A. Grimm, and Thomas W. Pendleton, *NASA Johnson Space Center, Houston, Texas* .... 421 -55

Terrestrial Applications of NASA Space Telerobotics Technologies
David B. Lavery, *NASA Headquarters, Washington, D.C.* .... 434 -56

On-Orbit Spacecraft Servicing - An Element in the Evolution of Space Robotics Applications
Carl J. Anders and Claude H. Roy, *Spar Aerospace Ltd., Brampton, Ontario, Canada* .... 441 -57

A Modular Artificial Intelligence Inference Engine System (MAIS) for Support of On Orbit Experiments
Thomas M. Hancock III, *New Technology Incorporated, Huntsville, Alabama* .... 451- [illegible]

Volume II
(March 23-24)

Robotic Sensing, Vision, and Perception Track: *Hatem Nasr, Chair*
Vision Systems Integration and Architecture Session: *Martial Herbert, Chair*

Research on an Autonomous Vision-Guided Helicopter
Omead Amidi, Yuji Mesaki, and Takeo Kanade, *Carnegie Mellon University, Pittsburgh, Pennsylvania* 456

Real-Time Tracking of Objects for Space Applications Using a Laser Range Scanner
Francois Blais, R. A. Couvillon, and Marc Rioux, *National Research Council/IIT, Ontario, Canada*; S. G. MacLean, *Canadian Space Agency, St-Hubert, Quebec, Canada* 464

Integration for Navigation on the UMass Mobile Perception Lab
Edward M. Riseman, Allen R. Hanson, Bruce Draper, and Benny Rochwerger, *University of Massachusetts at Amherst, Amherst, Massachusetts*; Claude Fennema, *Mount Holyoke College, Hadley, Massachusetts* 473

The 4-D Approach to Visual Control of Autonomous Systems
Ernst D. Dickmanns, *Universitaet der Bundeswehr Muncheon, Neubiberg, Germany* 483

Fusion of Laser and Image Sensory Data for 3-D Modeling of the Free Navigation Space
Mark Maas, N. G. Bourbakis, and A. Moghaddamzadeh, *Binghamton University, Binghamton, New York* 494

Contents
(continued)

Planning, Reasoning, and Control Track: *R. Peter Bonasso, Chair*

Selective Perception and Human Robot Interaction Session: *Avi Kak, Chair*

Tele-Assistance for Semi-Autonomous Robots
Robin R. Murphy, *Colorado School of Mines, Golden, Colorado*; Erika Rogers, *Clark Atlanta University, Atlanta, Georgia* 500

Mobile Robot Exploration and Navigation of Indoor Spaces Using Sonar and Vision
David Kortenkamp, *The MITRE Corp., Houston, Texas*; Marcus Huber, Frank Koss, Jaeho Lee, Annie Wu, William Belding, Clint Bidlack, and Seth Rodgers, *The University of Michigan, Ann Arbor, Michigan* 509

The Ground Vehicle Manager's Associate
Gary Edwards, Bruce L. Bullock, Gary R. Edwards, Robert H. Burnard, and William L. Bewley, *ISX Corporation, Westlake Village, California* 520

Deictic Primitives for General Purpose Navigation
Jill D. Crisman and Michael Clearly, *Northeastern University, Boston, Massachusetts* 527

A Methodology for the Generation of the 2-D Map from Unknown Navigation Environment by Traveling a Short Distance
N. G. Bourbakis and D. Sarkar, *Binghamton University, Binghamton, New York* 538

Systems Technology and Architectures Track: *Arthur C. Sanderson, Chair*

Robotic Systems Technologies Session: *Raymond W. Harrigan, Chair*

Simplifying Applications Software for Vision Guided Robot Implementation
Charlie Duncheon, *Adept Technology, Inc., San Jose, California* 543

An Open Architecture Motion Controller
Lothar Rossol, *Trellis Software & Controls, Inc., Rochester Hills, Michigan* 551

Telerobotics for Depot Modernization
Michael B. Leahy, Jr. and S. B. Petroski, *Kelly Air Force Base, San Antonio Air Logistics Center Kelly AFB, Texas* 557

A Smart Telerobotic System Driven by Monocular Vision
Rui J. P. deFigueiredo, Andrea Maccato, and Peter Wlczek, *University of California at Irvine, Irvine, California*; Bradley Denney and John Sheerer, *McDonnell Douglas Aerospace, Huntington Beach, California* 563

GA-Optimization for Rapid Prototype System Demonstration
Jinwoo Kim and Bernard P. Zeigler, *University of Arizona, Tucson, Arizona* 571

A Robot Control Formalism Based on an Information Quality Concept
Anders Ekman, Anders Torne, and Dan Stromberg, *FOA (National Defense Research Establishment), Linkoping, Sweden* 580

Contents
(continued)

Field Track: *Harvey B. Meieran, Chair*

Military and Other Field Applications Session: *John A. Davis, Chair*

The Network Data Delivery Service: A Real-Time Data Connectivity System
Stan Schneider, *Real-Time Innovations, Inc., Stanford University Aerospace Robotics Laboratory, Sunnyvale, California* 591

Robotics in A Controlled Environment Agriculture
Gaines E. Miles, *Purdue University, West Lafayette, Indiana*; Jon D. Erickson, *NASA Johnson Space Center, Houston, Texas* 598

Robotic Hauling Truck for Surface Mining
Keith Chrystall, Patrick Feighan, and Peter Wojcik, *Alberta Research Council, Calgary, Alberta, Canada*; Julian Coward, *Syncrude Canada Ltd., Edmonton, Alberta, Canada*; Ron Eirich and Clement Laforce, *Defence Research Establishment Suffield, Alberta, Canada* 606

A Nonlinear Strategy for Sensor Based Vehicle Path Control
Robert Mayr, *University of Southern California, Los Angeles, California* 614

The Problem with Multiple Robots
Marcus J. Huber and Patrick G. Kenny, *The University of Michigan, Ann Arbor, Michigan* 620

Factory Track: *Michael B. Leahy, Jr., Chair*

Dual-Use Precommercial Robotic Technology Session: *Francis daCosta, Chair*

A Generic Telerobotics Architecture for C-5 Industrial Processes
Wayne Zimmerman and Paul G. Backes, *NASA Jet Propulsion Laboratory, California Institute of Technology Pasadena, California*; Michael B. Leahy, Jr., *Kelly Air Force Base, San Antonio Air Logistics Center Kelly AFB, Texas* 630

A Practical Method of Reverse Engineering and Automatic Path Programming for Robotic Surface Finishing
William T. Adams, John M. Fitzgerald, T. J. Lawley, and O. R. Mitchell, *University of Texas at Arlington, Fort Worth, Texas* 640

Virtual Environments for Telerobotic Shared Control
Brian K. Christensen, *Deneb Robotics, 3285 Lapeer Road West P. O. Box 214687 Auburn Hills, Michigan 48321* 646

Designing the Next Generation of Robotic Controllers
David G. Goldstein, *North Carolina A&T State University, Computer Science Department, Greensboro, North Carolina 27411* 656

Service Track: *John M. Holland, Chair*

Building Operations Session: *Charles "Buck" Ward, Chair*

An End User's Wishlist
Charles W. Ward, *Commercial Service of Virginia, Inc., Richmond, Virginia* 662

Contents
(continued)

The First Commercial Floor Care Company That Ventured into the Production of Robotics
Allen J. Bancroft, *The Kent Company, Elkhart, Indiana* 669

Space Track: *Joseph C. Parrish, Chair*

Planetary Exploration Applications Session: *Brian H. Wilcox, Chair*

Non-Geometric Hazard Detection for a Mars Microrover
Brian H. Wilcox, *NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California* 675

Low Computation Vision-Based Navigation for a Martian Rover
A. S. Gavin, Rodney A. Brooks and Andrew S. Gavin, *Massachusetts Institute of Technology, Cambridge, Massachusetts* 685

The "MITY" Micro-Rover: Sensing, Control, and Operation
Eric Malafeew and William Kaliardos, *Massachusetts Institute of Technology, Cambridge, Massachusetts* 696

Supervised Space Robots Are Needed in Space Exploration
Jon D. Erickson, *NASA Johnson Space Center, Houston, Texas* 705

A Multitasking Behavioral Control System for the Robotic All Terrain Lunar Exploration Rover (RATLER)
Paul Klarer, *Sandia National Laboratories, Albuquerque, New Mexico* 717

Robotic Sensing, Vision, and Perception Track: *Hatem Nasr, Chair*

Vision Systems Integrations and Architecture II Session: *Martial Herbert, Chair*

SAVA III: A Testbed for Integration and Control of Visual Processes
James L. Crowley, *LIFIA-IMAG, Grenoble, France*; Henrik Christensen, *Aalborg University, Aalborg, Denmark* 724

An Architecture for Real-Time Vision Processing
Chiun-Hong Chien, *Lockheed Engineering & Sciences Company, Houston, Texas* 736

Motion Estimation of Objects in KC-135 Microgravity
Lisa Hewgill, *Lockheed Engineering & Sciences Company, Houston, Texas* 744

Real-Time Tracking of Objects for a KC-135 Microgravity Experiment
Mark L. Littlefield, *Lockheed Engineering & Sciences Company, Houston, Texas* 750

Grasping Objects Autonomously in Simulated KC-135 Zero-G
Robert S. Norsworthy, *Lockheed Engineering & Sciences Company, Houston, Texas* 757

Object Tracking with Stereo Vision
Eric L. Huber, *The MITRE Corporation, Houston, Texas*; Kenneth Baker, *NASA Johnson Space Center, Houston, Texas* 763

Contents
(continued)

Planning, Reasoning, and Control Track: *R. Peter Bonasso, Chair*

Planning Session: *R. Peter Bonasso, Chair*

A Software Architecture for Hard Real-Time Execution of Automatically Synthesized Plans or Control Laws
Marcel Schoppers, *Robotics Research Harvesting, Redwood City, California* 768

All Feasible Plans Using Temporal Reasoning
Debasis Mitra and Rasiah Loganantharaj, *University of Southwestern Louisiana, Lafayette, Louisiana* 776

Integrating Deliberative Planning In a Robot Architecture
Christopher Elsaesser and Marc G. Slack, *The MITRE Corporation, McLean, Virginia* 782

Planning in Subsumption Architectures
Eugene C. Chalfant, *University of Southern California, Los Angeles, California* 788

Real-Time Robot Deliberation by Compilation and Monitoring of Anytime Algorithms
Shlomo Zilberstein, *University of Massachusetts at Amherst, Amherst, Massachusetts* 799

Passive Mapping and Intermittent Exploration for Mobile Robots
Sean P. Engelson, *Yale University, New Haven, Connecticut* 810

Systems Technology and Architectures Track: *Arthur C. Sanderson, Chair*

New Directions in Robotic Systems Session: *Samad Hayati, Chair*

Extensibility in Local Sensor Based Planning for Hyper-Redundant Manipulators (Robot Snakes)
Howie Choset and Joel Burdick, *CalTech, Pasadena, California* 820

Fault Tolerant Kinematic Control of Hyper-Redundant Manipulators
Nazareth S. Bedrossian, *C. S. Draper Laboratory, Houston, Texas* 830

Failure Tolerant Operation of Kinematically Redundant Manipulators
Anthony A. Maciejewski and Christopher L. Lewis, *Purdue University, West Lafayette, Indiana* 837

UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications
Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Patrick G. Kenny, *The University of Michigan, Ann Arbor, Michigan* 842

Control of Parallel Manipulators Using Force Feedback
Prabjot Nanua, *University of Houston, Houston, Texas* 850

Robust Inverse Kinematics Using Damped Least Squares with Dynamic Weighting
D. E. Schinstock, T. N. Faddis, and R. B. Greenway, *University of Kansas, Lawrence, Kansas* 861

Contents
(concluded)

ControlShell: A Real-Time Software Framework
Stan Schneider, Vincent Chen, and Gerardo Pardo-Castellote, *Real-Time Innovations, Inc., Stanford University Aerospace Robotics Laboratory, Sunnyvale, California* 870

Second Plenary Session: *George Kozmetsky, Chair*
Commercialization

JSC Proposed Dual-Use Technology Investment Program in Intelligent Robotics
Jon D. Erickson, *NASA Johnson Space Center, Houston, Texas* 878

Monday
March 21, 1994

ꝏ?22 **AIAA-94-1301-CP**

INTELLIGENT ROBOTICS CAN BOOST AMERICA'S ECONOMIC GROWTH

/.//

Jon D. Erickson*
National Aeronautics and Space Administration Lyndon B. Johnson Space Center
Houston, Texas

5/-63

N94-30527

Abstract

A case is made for strategic investment in intelligent robotics as a part of the solution to the problem of improved global competitiveness for U.S. manufacturing, a critical industrial sector. Similar cases are made for strategic investments in intelligent robotics for field applications, construction, and service industries such as health care. The scope of the country's problems and needs is beyond the capability of the private sector alone, government alone, or academia alone to solve independently of the others. National cooperative programs in intelligent robotics are needed with the private sector supplying leadership direction and aerospace and nonaerospace industries conducting the development. Some necessary elements of such programs are outlined.

The National Aeronautics and Space Administration (NASA) and the Lyndon B. Johnson Space Center (JSC) can be key players in such national cooperative programs in intelligent robotics for several reasons: (1) human space exploration missions require supervised intelligent robotics as enabling tools and, hence, must develop supervised intelligent robotic systems; (2) intelligent robotic technology is being developed for space applications at JSC (but has a strong crosscutting or generic flavor) that is advancing the state of the art and is producing both skilled personnel and adaptable developmental infrastructure such as integrated testbeds; and (3) a NASA JSC Technology Investment Program in Robotics has been proposed based on commercial partnerships and collaborations for precompetitive, dual-use developments.

*Chief Scientist, Automation and Robotics Division, Member AIAA

1. Introduction

Intelligent robotics can boost America's economic growth by enabling productivity improvements that raise the standard of living for everyone and by enabling the U.S. to build products at world cost and quality.[1] But the boost can occur only if intelligent robotics technology is developed as a mature commercial capability and is used to solve productivity problems in critical sectors of the economy (e.g., advanced manufacturing, construction, field applications, and service industries such as health care). Both the development of intelligent robotic systems and their early application in these strategic sectors require strategic investment. The government, and NASA in particular, should contribute to the strategic investment by buying down the risk for commercial technology. The government can accomplish this by developing needed intelligent robotic systems for government applications and then sharing the technology with the commercial sector in ways that allow profitable products. These precommercial, dual-use investments and developments are in line with President Clinton's technology policy.[2]

Intelligent robotics is the use of robotic systems in solving problems in tasks and environments where the robot's ability to acquire and apply knowledge and skills to achieve stated goals in the face of variations, difficulties, and complexities imposed by a dynamic environment having significant unpredictability is crucial to success. This means the robots can recognize and respond to their environments at the pace of their environments and to spoken human supervision in order to perform a variety of mobility and manipulation tasks. This does not require a broad-based general intelligence or common sense by the robot.

These robots are capable of significant autonomous reaction to unpredictable events, yet they are subject to optional human supervision during operation in a natural way such as by voice. We refer to this capability in the supervised robot as "adjustable autonomy."

I believe that the most important path to fundamental change in the U.S. economy is a long-term focus on actions that will provide strategic investment in our nation's future. I believe that investments in intelligent robotics-related innovations of the precommercial, dual-use variety will lead to products that are ready to be commercialized and introduced into the marketplace, which, in turn, can provide a valuable solution to at least a part of our continuing economic crises.

A lack of foresight in this area could inhibit American competitiveness in today's and tomorrow's global economy.

Intelligent robotic systems mean that less structuring of the robot environment is required to obtain robotic task performance, which, in turn, means lower costs. For those applications where structuring the environment is generally not possible, intelligent robotics offers the flexibility to enable robotic tasks otherwise not possible. Packaging mobility with manipulation as intelligent robotics allows frequently means fewer manipulators than otherwise, further lowering costs.

The benefits of innovation transcend the new technologies themselves. Because new technology allows more cost-effective investment in infrastructure and commercial competitiveness, the U.S. will be more competitive globally. This, in turn, will produce more jobs, improve the economy, and reduce the trade and budget deficits.

The scope of the country's problems and needs is far beyond the capability of the private sector alone, government alone, or academia alone. Cooperation by all interested parties is essential to maximize the benefits from innovation! National cooperative programs are needed with leadership direction from the private sector.

To support this approach, an example of a cooperative program currently ongoing is the University Space Automation and Robotics Consortium (USARC) consisting of the University of Texas at Austin, Texas A&M University, Rice University, the University of Texas at Arlington, MITRE Corporation, and NASA JSC.

However, all this is of major interest only because intelligent robotics is within our reach as a commercial technology, although perhaps not yet within our grasp. Major intelligent robotics capabilities exist in many places in industry (e.g., Transitions Research Corporation, Teleos, Sarcos, Robotics Research Harvesting, and Intellagent Systems), in not-for-profit companies (e.g., MITRE Corporation, SRI International, and Southwest Research), in academia (e.g., Carnegie Mellon University, Massachusetts Institute of Technology, Stanford University, University of Michigan, Rensselaer Polytechnic Institute, University of Pennsylvania, and USARC), and in government (e.g., NASA, National Institute of Standards and Technology, and Department of Energy). Many intelligent robotics efforts have been reported.[3] These organizations and activities could form the basis of cooperative national programs that could pay off in the near term.

2. Investment for U.S. Manufacturing

"Both American industry and government under-invest in manufacturing. In contrast to their foreign competitors, U.S. firms neglect process-related R&D within their overall R&D portfolio. And the federal government allocated only two percent of its $70 billion R&D budget to manufacturing R&D in FY92."[2]

Strategic investment in and use of intelligent robotics in manufacturing offer partial solutions to many cost and quality problems, if the robotic systems are properly targeted and designed. Product manufacturers and their people must identify problem areas and ways to integrate intelligent robotics into their manufacturing processes. Flexible approaches by robot manufacturers are necessary to offer solutions to problems rather than robots per se. These problem identification and proposed solution activities both require strategic investments that government should help with in order to buy down the initial risk. Leading technology needs are in sensors and information extraction techniques from sensor data to support the task needs.

One of the needed developments is to reduce the cost of production of the intelligent robots themselves through generic software architectures (standardized and modular) and modular hardware approaches.

The benefits to product manufacturing of such strategic investment are as follows:

- Having the ability to build products at world cost and quality
- Improving productivity
- Reducing time to market for manufactured products

- Reducing costs
- Improving quality
- Improving our global competitiveness
- Having the ability to preserve and create jobs in manufacturing
- Creating jobs in intelligent robotics for manufacturers and integrators including training and support
- Improving the economy and boosting economic growth
- Increasing profits
- Increasing tax revenue
- Reducing the trade deficit
- Reducing the budget deficit
- Raising the standard of living for everyone

A cost-benefit analysis for intelligent robotics in manufacturing should be conducted if this will help make the case more compelling for all parties.

Also, manufacturing is a strategic industry related to defense and national security in non-threatening ways, so that its vitality is not simply a pure economic issue.[4]

3. Investment for Applications in Other Sectors

Cases for strategic investment in intelligent robotics for applications in other sectors have just as compelling a basis of rationale and benefits as manufacturing, whether in construction, mining, agriculture, undersea applications, health care, nuclear power, or other service applications such as grocery warehouse uses.

In construction, both the national architectural and engineering firms and the civil engineering community want more productive methods such as those offered by intelligent robotics.[5,6] Our physical infrastructure is deteriorating at an exceedingly dangerous rate.[7] This includes our highways and bridges, mass transit, aviation facilities, water transportation, waste-water treatment, drinking water distribution systems, and a host of other public works and public facilities. This is the physical framework that supports and sustains virtually all domestic economic activity; it is essential to maintaining international competitiveness as well. Intelligent robotics applications could reduce the cost of replacing or upgrading much of this infrastructure.[6] NASA, likewise, needs space construction intelligent robotics.

Intelligent robotics is required in mining in order to enable the U.S. to remain globally competitive cost-wise in coal production, and to improve mine safety for miners.[8] Clearly energy and its cost are fundamental to industrial competitiveness in the global economy. The deeper coal veins, in general, do not have large cross-sectional areas at the coal interface, and robotics that can sense the vein edges from the surrounding rock are needed. Transportation to the surface is another task where robotics could aid productivity. Again, NASA needs mining intelligent robotics for large-scale planetary resource use.

In controlled environment agriculture, which is a several billion dollar per year business in the U.S., intelligent robotics is needed to keep prices competitive.[9] Market forces are compelling greenhouse operations, which are labor intensive, to automate. A major motivation is for U.S. producers to improve productivity in international competition.[10] Similarly, NASA needs intelligent robotics in advanced life support systems where higher order plants (crops) will be used in food production, water purification, carbon dioxide uptake, and oxygen release as part of the bioregenerative recycling systems that need little, if any, resupply.[11]

A large number of other sectors and applications of intelligent robotics is evident, from undersea applications to nuclear power and a number of service industry uses.[12] Low-cost health care is another critical factor in global competitiveness as a major labor-related cost, and intelligent robotics can reduce costs while increasing quality. Despite the varied capabilities of current field and service robots, there are many additional tasks awaiting future field and service robots. Some robots will be cleaning up toxic and radioactive waste and monitoring water pollution. Other robots will provide mobility aids for the handicapped and infirm and bring new forms of education and entertainment. The time required to add these capabilities is measured not in years but in person years of research and development.

4. Cooperative Programs in Intelligent Robotics

In this section we describe some necessary elements of cooperative national programs in intelligent robotics. This section is based to a significant extent on Carlisle.[13]

First, we must communicate a sense of urgency about the critical importance of manufacturing technology to our country's executives, financial community, and government. Our cost of labor will not likely compete with Singapore or Mexico. But Japan, whose cost of labor is equal to ours, has shown that it is possible to build products at world cost and quality through the use of automation technology. Our government and our boards of directors are asking the question, "What is the manufacturing strategy that will keep us competitive in the world market and will retain jobs?"

Second, we need a manufacturing and automation technology education infrastructure. President Clinton has proposed establishing 170 technology extension centers where local businesses can learn about new technology on state-of-the-art machinery.[2] The Robotics Industries Association (RIA) is developing an encyclopedia of robot applications that, combined with equipment at these technology centers, could greatly accelerate the adoption of robot technology by U.S. industry. Another education-related activity involves communication of information about intelligent robotics and concurrent engineering. JSC is involved in the National Information Infrastructure Testbed (NIIT), which is an industry-led consortium to initiate the "information superhighway," where the government role is primarily to conduct needed research and development and determine the policy environment and legal situation. But another key government role in NIIT that concerns us at JSC is providing technology information, both about intelligent robotics and about concurrent engineering, over the Internet. NIIT members include AT&T, Sprint, Hewlett-Packard, Digital Equipment Corporation, SynOptic Communications, Sun Microsystems, Ellery Systems, Novell, U.S. West Communications, New England T&T, Sandia National Laboratories, University of New Hampshire, Oregon State University, University of California, and Ohio State University. The JSC activity involves providing access to information on intelligent robotics via the Internet and using the Internet as a distributed computing environment for access to a suite of interoperable engineering software applications that support a structured process for concurrent engineering.[14]

Third, the cost and availability of capital for productive investment must be addressed. Japan is providing more than 20 times the amount of federally guaranteed loans to small business than the U.S. is providing – $80 billion per year in Japan versus $3.6 billion in 1989 in the U.S. Also, Japan provides tax credits and zero percent interest loans up to $0.25 million for mechatronics equipment. Banks in the U.S. are still extremely hesitant to make loans to small and midsize businesses due to regulatory pressure as a result of the savings and loan collapse. We need to improve and encourage productive private investment through changed banking regulations and tax policies.

Finally, we must encourage applied research and development on robotic systems for field, construction, factory, service, and space applications. There has been almost no U.S. research funding for industrial applications where we need it most to help us compete in quality and cost in the global market. Nor has there been funding for construction applications where rebuilding our infrastructure is a needed major strategic investment. We need to direct funds toward developing practical applications of robotic systems as integrated solutions to industrial productivity problems. We need to develop system testbeds such as JSC has developed where developers can integrate sensing, control, and mechanical technologies with the objective of testing robotic solutions to actual industrial applications.

5. Johnson Space Center Role

NASA and JSC can be key players in national cooperative programs in intelligent robotics for several reasons: (1) human space exploration missions require supervised intelligent robotics as enabling tools and, hence, must develop or have developed supervised intelligent robotic systems;[15] (2) intelligent robotic technology is being developed for space applications at JSC (but has a strong crosscutting or generic flavor) that is advancing the state of the art and is producing both skilled personnel and adaptable developmental infrastructure such as low-cost simulation environments for software testing and integrated testbeds for complete prototype testing;[16, 17] and (3) a NASA JSC Technology Investment Program in Robotics has been proposed based on commercial partnerships and collaborations for precompetitive, dual-use developments.[18] The JSC Technology Investment Program suggests efforts on generic intelligent robotics software architectures, modular manipulation and mobility designs, integrated sensing and perception, dexterous grasping and manipulation, and prototyping and rapid development environments, all as part of an approach for end-user customizing of intelligent robotic systems. The JSC Technology Investment Program also suggests problem-solving approaches to applications in several sectors. JSC also has a Small Business Innovative Research (SBIR) program for intelligent robotics, which is underutilized and has no commercial cost sharing requirement.[19] It is limited in scope to about $0.6 million and 2 years in Phase II efforts.

A key element in the cutting edge intelligent robotics technology work at JSC is an understanding of and solution approach to the key issue of melding artificial intelligence planners with reactive capabilities. Artificial intelligence planners offer goal-achieving planning, but also high-time variance due to searching. Reactive capabilities are needed to deal safely in real time with dynamic, unpredictable environments at the pace of the dynamics[16]. A second key element that JSC brings is an approach to improved robotic reliability as required for space, but also useful in industry. A third key element that JSC brings to cutting edge technology is an understanding of and solution approach to the key issue of robotic safety while maintaining productivity.

Of all of these elements, the most important one is the personnel skilled in the state of the art and knowledgeable about the technology.

6. The Role of Government

The proper role of government in industry, in general, and intelligent robotics, in particular, may be controversial. Government establishes the environment within which business operates such as laws, taxes, and services. Government provides education and training funding and negotiates mutual trade policy such as the North American Free Trade Agreement (NAFTA). Government also spends $70 billion per year on research and development.

The global competitive landscape may be different today than we have assumed in America. Peter Drucker points out: "The emergence of new non-Western trading countries – foremost the Japanese – creates what I would call adversarial trade. ... Competitive trade aims at creating a customer. Adversarial trade aims at dominating an industry. ... The aim in adversarial trade ... is to drive the competitor out of the market altogether rather than to let it survive."[20]

James Fallows argues about the semiconductor industry: "The prevailing American idea requires us to view industrial rises and falls as if they were the weather. We can complain all we want, but in the long run there's nothing much we can do, except put on a sweater when it's cold. Or the American idea makes economic change seem like an earthquake: some people are better prepared for it than others, but no one can constrain the fundamental force. A different idea – that industrial decline is less like a drought than like a disease, which might be treated – would lead to different behavior."[21]

"On its way up and on its way down, the semiconductor industry was driven not just by private companies – although they made every crucial operating decision and came up with every new design – but by a network of government-business interactions."[21]

Fallows quotes the Semiconductor Industry Association: "Government policies have shaped the course of international competition in microelectronics virtually from the inception of the industry, producing outcomes completely different than would have occurred through the operation of the market alone."[21]

Again Fallows states: "For instance, in 1962 NASA announced that it would use integrated circuits – the first simple chips, produced by Texas Instruments, Fairchild Semiconductor, and other suppliers – in the computer systems that would guide Apollo spacecraft to the moon. ... Every history of the semiconductor business regards these contracts as a turning point; they guaranteed a big and relatively long term market, which no private purchaser could have offered at the time ... price went down, and commercial customers began buying more and more chips. ... Government contracts had paid for some of the research that led to patents."[21]

"For aircraft ... even more than with semiconductors, the government provided the initial market. ... Governments may not be able to pick winners, but they seem to be able to make winners."[21]

The precompetitive, dual-use technology investment concept advocated here for intelligent robotics appears to have many successful historical precedents in buying down initial commercial risk and attracting commercial development.[22]

7. Conclusion

We have the intellect and skill in the U.S. to make use of intelligent robots in ways that will boost our economic growth, greatly improve our national ability to compete in the global economy through advanced manufacturing at world cost and quality, create jobs in manufacturing of intelligent robots, improve the quality and reduce the cost of health care, provide needed cost reductions and productivity improvements in construction and mining, and, in fact, preserve manufacturing in the U.S. What we lack, perhaps, is the perception and commitment that this is a strategy we must pursue. Our Congressional track record is less than promising in investing in robotics, but there are signs of hope.[23] We need the commitment of an Andrew Rowan taking "A Message to Garcia," as opposed to "letting someone else do it."[24]

We are at a stage of developing intelligent robotics where a major cooperative development effort would pay off in the near term – less than 5 years, rather than 10 years or more; in fact, the metric should be in person years, not calendar years.

JSC and its Automation and Robotics Division stand ready with intelligent robotics technology, skilled people, low-cost simulation approaches and integrated robotic testbeds, a suggested set of activities for commercial involvement in partnerships, matching funding possibilities, and a small business innovative research program that does not require any cost sharing.

Industry must step forward and lead, but NASA should do its part in supporting development by industry through technology sharing and providing some risk reducing investment funding.

8. References

1. Office of Technology Assessment: "Exploring the Moon and Mars: Choices for the Nation." Congress of the U.S., Washington, DC, Aug.1991.

2. Clinton, William J.; and Gore, Jr., Albert: "Technology for America's Economic Growth, A New Direction to Build Economic Strength." White House, Washington, DC, Feb. 22, 1993, pp.1-35.

3. IEEE Transactions on Systems, Man, and Cybernetics: "Special Issue on Unmanned Vehicle and Intelligent Robotic Systems." vol. 20, no. 6, Nov. 1990.

4. Bureau of Export Administration: "National Security Assessment of the U.S. Robotics Industry." Department of Commerce, Washington, DC, Apr. 1991.

5. Yates, R. E.: "Not Level With Competition." Chicago Tribune Business Section, Nov. 9, 1992, p. 1.

6. Civil Engineering Research Foundation: CERF Currents, vol. 93, no.1, Washington, DC, winter 1993, p.5 .

7. Haimes, Y. Y.: "Options for National Infrastructure Renewal." IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 4, July 1991, p.701.

8. Schnakenberg, Jr., G. H.: "How Computers Can Make Coal Mining Safer and More Competitive." Minerals Today, Dec. 1992, p.12.

9. Simonton, W.: "Automatic Geranium Stock Processing in a Robotic Workcell." Transactions of American Society of Agricultural Engineering, vol. 33, no. 6, Nov.-Dec. 1990, pp. 2074-2080.

10. Carney, L.: Oglevee Products, McDonough,GA, 1988, as cited by Simonton above.

11. Miles, G. E.; and Erickson, J. D.: "Robotics in Controlled Environment Agriculture." Proceedings of AIAA-NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

12. Engelberger, Joseph F.: "Robotics in Service." The MIT Press, Cambridge, MA, Jan. 1991.

13. Carlisle, B.: "A National Plan for Robotics." Robotics World Magazine, vol.11, no. 1, Mar. 1993, p.4.

14. Erickson, J.D.; and Lawler, D.: "Integrated Computer Aided Concurrent Engineering." Dual-Use Space Technology Transfer Conference and Exhibition, NASA Johnson Space Center, Houston, TX, Feb. 1994.

15. Erickson, J. D.: "Supervised Space Robots Are Needed in Space Exploration." Proceedings of AIAA-NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

16. Erickson, J .D.; et al.: "An Intelligent Space Robot for Crew Help and Crew and Equipment Retrieval." International Journal of Applied Intelligence, accepted for publication in 1994

17. Erickson, J D., Grimm, K. A.; and Pendleton,T. W.: "An Intelligent Robot for Helping Astronauts." Proceedings of AIAA-NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

18. Erickson, J. D.: "JSC Proposed Dual-Use Technology Investment Program in Intelligent Robotics." Proceedings of AIAA-NASA Conference.on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

19. NASA, Small Business Innovation Research Program Solicitation, SBIR 93-1, NASA Headquarters, Washington, DC, May 1993.

20. Drucker, P. F.: "The New Realities." Harper and Row, New York, NY, 1989.

21. Fallows, J.: "Looking at the Sun." The Atlantic Monthly, Nov. 1993, p.69.

22. Brown, H.; et al.: "The Government Role in Civilian Technology – Building a New Alliance," National Research Council, National Academy Press, Washington, DC, 1992.

23. NASA Advanced Technology Advisory Committee: "Advancing Automation and Robotics Technology for the Space Station Freedom and for the U.S. Economy." NASA Tech Memo semi-annual progress reports to Congress since 1984.

24. Hubbard, Elbert: "A Message to Garcia." 1899.

N/S
52-63
207025
p-4

6023

AIAA-94-1176-CP

N94-30528

TELEOPERATED SYSTEMS FOR NUCLEAR REACTORS INSPECTION AND MAINTENANCE

V.P.Dorokhov
Engineering Firm "TECHNIKA"
St.-Petersburg, Russia

D.V.Dorokhov
Baltic State Technical University
St.-Petersburg, Russia

A.P.Eperin
Leningrad Atomic Power Station
Sosnoviy Bor, Russia

Abstract

The present paper describes author's work in the field of teleoperated equipment for inspection and maintenance of the RBMK technological channels and graphite laying, emergency operations. New technological and design solutions of teleoperated robotic systems developed for Leningradsky Power Plant are discussed.

1. Introduction

This paper is one from the series devoted to nuclear power plant reliability and safety improvement with the help of teleoperated and automatic manipulative systems providing inspection and maintenance of RBMK reactor channels. The same robotic systems could be implemented in case of severe accidents at nuclear energy objects and for other technical applications (chemical industry, space, military technologies, etc.)

Main components of the system under development:

- robot for fuel assemblies handling;
- advanced teleoperated / automatic sensor-based manipulator for the reactor hall;
- teleoperated / automatic mobile manipulator for the under reactor zone;
- remote inspection system for technological channels;
- technological manipulative system for graphite laying repair;
- underwater robotic system for the nuclear fuel storage pool;
- remote inspection system for pipes and tubes of the first reactor loop diagnostic;
- mono and stereo TV systems;
- heavy duty crane for the central hall.

All these remotely controlled systems could be considered as cybernetic environment (under the human operator supervising) providing inspection, maintenance and emergency operations in the central hall, under-reactor zone and inside the technological channels and cells of the reactor.

2. Object for robots implementation

Two working zones are considered: the central (reactor) hall and reactor itself. The central hall situated above the reactor is 24 m wide, 54 m long and 33 m in height.

In fact the floor of the central hall is an object of maintenance. The most probable task is removing of fuel assembly tablets parts, which can fall down to the central hall floor during process of transportation to the storage pool. Radiation situation in this case depends upon quantity of the lost fuel and distance from the manipulator.

There are 2 main difficulties in reactor technological channel inspection and maintenance: strictly limited geometrical parameters of the channel and very high level of radioactivity. A technological RBMK channel is a vertical tube of complex geometry made of Zr and Nb alloy with minimum internal diameter - 80 mm and total length - 18 m. Active reactor zone length - 7m. Technological channels are surrounded with graphite laying which consists of graphite blocks (rectangular, 250x250 mm with a hole of 114 mm in diameter). A fuel assembly is placed hermetically inside the channel. Radioactivity inside graphite blocks and channels of stopped reactor comprises alfa, beta and gamma-rays. At Leningradsky power plant reactor channels gamma-rays level is about 1800-2000 roentgen per hour.

3. Proposed solutions

Channel type reactor maintenance practice determined developing of 3 independent teleoperated systems for inspection and remote handling:

1. Teleoperated / automatic robot for the central hall providing a wide range of technological operations under unpredictable environmental conditions.

2. Remotely controlled inspection and fuel assembly pieces removal system for the channel with cross section diameter of 80 mm and length of 18 m.

3. Teleoperated system for handling and removal of fuel assembly and graphite blocks pieces from the channel with rectangular cross section (250x250 mm).

Teleoperated systems should include several subsystems such as: TV viewing system, lighting system, manipulator, geometric parameters measuring system, gamma-rays level measuring system, end effector fixation system, azimuth measuring system.

4. Design implementation

Bellow follows a brief description of the above mentioned teleoperated systems.

Teleoperated / automatic robot

Teleoperated / automatic robot consists of remote master-slave manipulator; stereo-TV system; mono-TV system (3 pieces); transport device, special set of changeable tools, control and operation equipment.

Manipulator consists of articulated slave arm with joint drive units containing electric motors, harmonic gears, speed, position and advanced torque transducers (the slave is made of titanic alloys and stainless steel), replica master arm, equipped with position transducers and brushless DC motors, control console containing standard electronics housing cages, operator console, cable set, set of tools. The slave drives contains 5 bilateral drive systems with brushless DC motors, rectifiers and transistor invertors. Control system of manipulator provides teleoperated and automatic work regimes.

Main technical data:

max. load capacity, kg	25
degrees of freedom, number	5
gripper squeezing force range, N	50-600
max. distance between master and slave, m	100
total consuming power, kw	2.5
mass, kg:	
slave arm	90
master arm	70

Main design principles: bilateral servodrives with automatic force control and advanced force reflection, modular drive units (M-54 design principle), remotely changeable tools, ability to be placed at any of vehicles.

Remotely controlled channel inspection system

Main technical data:

TV camera rotation speed, deg/s	16
TV camera rotation angle, deg	360
mirror rotation angle, deg	45
mirror movement control	incremental
gripper linear movement range, mm	0 - 49
gripper load capacity, kg	0.09
max. distance between control console and manipulative system, m	50
gripper squeezing speed, 1/s	1
lifter max load capacity, kg	22
manipulative system mass, kg	15
lifter position accuracy, mm	10
lifting speed, mm/s	11 and 21

This teleoperated system contains a mobile module and a remote operator control console. The mobile module provides working operations inside the reactor channels and consists of a manipulative system and a tower with an automatic lock (for connecting to the central hall heavy duty crane), a lifter with a cable drum and a mechanism for accurate manipulative system positioning above a channel.

The manipulative system is designed in the form of a cylinder with cross section diameter of 49 mm with built-in:

- electromechanical gripper;
- rolling mirror;
- TV camera with an objective, lighting system and image processing equipment;
- gripper drive;
- TV camera rotation drive;
- azimuth movement drive;
- channel geometric parameters measuring system and manipulator fixation system;
- electrical connector.

The gripper situated at the bottom side of the manipulative system provides small objects removal out of the channel. The gripper comprises spring loaded tongs activated by an electric drive. (Squeezing force - 0.9 kg; maximum load capacity - 90 g).

Strict geometrical requirements made the designers solve a very complicated problem of all the parts of the system mounting consequently inside the narrow steel tube body. For example the system TV camera should provide both side and axial viewing modes. Therefore the inclined mirror should have 2 fixed positions. For this a special "mirror on/off" mechanism was developed. Its main idea is to implement the same motor for activating the gripper and rolling the mirror. This motor activates the screw which makes a nut with a cam slot on the surface move along the screw. While the mirror finger is in the vertical part of the slot the mirror remains in the same position. But when passing the sloping part of the slot the finger makes the mirror move opening the axial view for the camera. Moving the nut farther with the mirror finger sliding along the second vertical part of the slot the motor closes the gripper without changing the mirror position. Moving the nut backwards the motor opens the gripper and then can change the mirror position returning to side viewing mode.

Such design solution provides a considerable system dimensions reduction.

Another peculiar feature of this manipulative system is a combined mechanism for a channel diameter measuring and the system fixation inside the channel.

The mechanism comprises 3 metal (3,969 mm in diameter) balls built into the system body. The balls can partly move out of the body until they meet the channel walls and fix the device inside the channel. The rotating ring pushing the balls outside is connected to the motor activating the mechanism and to the position transducer providing accurate (+/- 0,001 mm) displacement and therefore channel diameter measurement.

Teleoperated manipulative system for channel graphite laying repair

This system should provide following technical tasks:

- internal graphite block viewing inspection;
- geometrical parameters measurement;
- sampling;
- channel, fuel assemblies and block parts removal;
- inside-cell and cell-to-cell blocks rearrangements.

The system should comprise:

- teleoperated manipulator with 200N load capacity;
- TV viewing system;
- measuring system;
- temperature sensors;
- container for small objects;
- grinding machine;
- changeable grippers.

The system working zone cross section varies from a circle of 114 mm in diameter to a square of 250x250 mm.

The system is currently under development.

5. References

1. V.P.DOROKHOV, D.V.DOROKHOV, A.F. NECHAEV "Remote systems: potential of application in uncertain conditions of hostile environments," Proc. of the ANS Fifth Topical Meeting on Robotics and Remote Systems, Knoxville, Tennessee / April 25-30, 1993.

2. V. P. DOROKHOV, "Master-slave remote control manipulator," Proc. of 1st International Symposium on theory and practice of robots and manipulators, Udina, Italy (1973).

3. G. KOHLER, Tupenbuch der manipulatoren. Verlag Karl Thiemig, Munchen (1985).

4. S. M. SHAKER, A. R. WISE, War without men (Robots on the future batterfield), Washington, Pergamon-Brossey's (1988).

5. M. KLARK, M. BRONE, "Telerobotics for the Space Station," Mechanical Engineering, Feb. (1986).

6. V. P. DOROKHOV, O. A. SADOVSKIY, "Servodrive with force reflection," Avt. svid. .N 317039, BI N 30 (1971).

7. V. P. DOROKHOV, "Remotely controlled balanced master-slave manipulator," Avt. svid. N415155, BI N6 (1974).

8. V. P. DOROKHOV, N. A. LAKOTA, "Peculiarities of the master-slave manipulators design," Proc. of the 6th all union Symposium on theory and design principles of robots and manipulators," Toliytti, (1976).

9. A. P. DOROKHOV, V.P.DOROKHOV, N.A.LAKOTA, "Servodrive with force reflection," Avt.

svid. N 643830, BI N 3 (1979).10. V. P. DOROKHOV, "Conception of unification of teleoperated manipulators," Proc. of the all union conference on practice, unification and standardization of industrial robots.

11. V. P. DOROKHOV, Fundamentals of robotics, Educational notebook, Part 1, Leningrad (1985).

12. V. P. DOROKHOV, Fundamentals of robotics, Educational notebook, Part 2, Leningrad (1986).

13. V. P. DOROKHOV, "Arm Servosystems using A.C.Motors", "Manipulator control Mechanisms," Remotely Controlled Robots and Manipulators, Mir Publishers, Moscow (1988).

14. D. V. DOROKHOV, "Servodrive with force reflection," Avt. svid. N 1531070, BI N 47 (1989).

15. V. P. DOROKHOV, V. I. MOKRUSHIN, L. I. NIKOLAEVA, "Computer aided design of induction motor based electrodrives," Preprint, Moscow, CNIIAtominform. (1987).

16. V. P. DOROKHOV, "Modern conception of nuclear robotics development," Proc. of the Seminar on robotic systems for complex automatization of nuclear energetics, St.-Petersburg, DNTP (1988).

53-63 N15 6024

p. 9

AIAA-94-1177-CP

N94- 30529

ARK: AUTONOMOUS MOBILE ROBOT IN AN INDUSTRIAL ENVIRONMENT

S.B. Nickerson[3], P. Jasiobedzki[1], M. Jenkin[2], A. Jepson[1], E. Milios[2], B. Down[1], J. R. R. Service[3], D. Terzopoulos[1], J. Tsotsos[1], D. Wilkes[3], N. Bains[4], T. Campbell[4]

[1] Dept. of Computer Science, University of Toronto, Canada M5S 1A4

[2] Dept. of Computer Science, York University, North York, Canada M3J 1P3

[3] Ontario Hydro Technologies, Toronto, Canada

[4] Atomic Energy of Canada Ltd., Mississauga, Canada L5K 1B2

Abstract

This paper describes research on the ARK (Autonomous Mobile Robot in a Known Environment) project. The technical objective of the project is to build a robot that can navigate in a complex industrial environment using maps with permanent structures. The environment is not altered in any way by adding easily identifiable beacons and the robot relies on naturally occurring objects to use as visual landmarks for navigation.The robot is equipped with various sensors that can detect unmapped obstacles, landmarks and objects. In this paper we describe the robot's industrial environment, it's architecture, a novel combined range and vision sensor and our recent results in controlling the robot, in the real-time detection of objects using their colour and in the processing of the robot's range and vision sensor data for navigation.

1. Introduction

The ARK (Autonomous Robot for a Known Environment) Project is a precompetitive research project involving Ontario Hydro, the University of Toronto, York University, Atomic Energy of Canada Ltd., and the National Research Council of Canada. The project started in September 1991 and will be completed in August 1995. The technical objective of the project is to develop a sensor-based mobile robot that can autonomously navigate in a known industrial environment.

There are many types of industrial operations and environments for which the mobile robots can be used to reduce human exposure hazards, or increase productivity. Examples include inspection for spills, leaks, or other unusual events in large industrial facilities, materials handling in computer integrated manufacturing environments, and the carrying out of inspections, the cleaning up of spills, or the carrying out of repairs in the radioactive areas of nuclear plants - leading to increased safety by reducing the radioactive dose to workers.

The industrial environment is significantly different from office environments in which most other mobile robots operate. The ARK project will produce a self-contained mobile robot with sensor-based navigation capabilities specifically designed for operation in a real industrial setting. The ARK robot will be tested in the large engineering laboratory at AECL CANDU in Mississauga, Ontario (figure 1). This open area covers approximately 50,000 sq. feet of space and accommodates one hundred and fifty employees. Within the Laboratory, there are test rigs of various sizes, mock-ups of reactor components, a machine shop, a fabrication facility, metrology lab and assembly area. There are no major barriers between these facilities and therefore at any one time there may be up to fifty people working on the lab floor, three fork lift trucks and floor cleaning machines in operation. Such an environment presents many difficulties that include: the lack of vertical flat walls; large open spaces (the main isle is 400' long) as well as small cramped spaces; high ceilings (50'); large windows near the ceiling resulting in time dependant and weather dependant lighting conditions, a large variation in light intensity, also highlights and glare; many temporary and semi-permanent structures; many (some very large) metallic structures; people and forklifts moving about; oil and water spills on the floor; floor drains (which could be uncovered); hoses and piping on the floor; chains hanging down from above, protruding

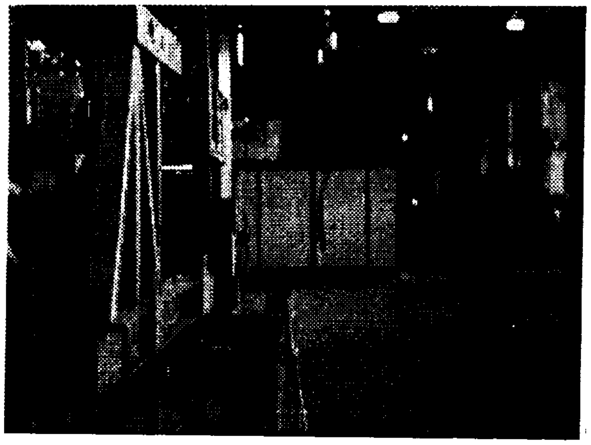


Figure 1. A view of the AECL industrial bay

structures, and other transient obstacles to the safe motion of the robot [11].

Large distances, often encountered in the industrial environment, require sensors that can operate at such ranges. The number of visual features (lines, corners and regions) is very high and techniques for focusing attention on specific, task dependent, features are required. Most mobile robotic projects assume the existence of a flat ground plane over which the robot is to navigate. In the industrial environment this ground plane is generally flat, but regions of the floor are marked with drainage ditches, pipes – this requires sensors that can reliably detect such obstacles.

The ARK robot's onboard sensor system consists of sonars and one or more ARK robotic heads and a floor anomaly detector (FAD). The head consists of a colour camera and a spot laser range finder mounted on a pan–tilt unit [5] (see also figure 3). The pan, tilt, camera zoom, camera focus and laser distance reading of the ARK robotic head are computer controlled. The ARK project is investigating different technologies for Floor Anomaly Detection (FAD) to detect objects on the floor that cannot be detected by the sonar system and are too large for ARK to traverse. One technology that is being developed is a laser based system built around the NRC BIRIS laser head[1]. A second approach is to use stereo vision to localize potential floor anomalies. Unlike the classical approach to stereo, the stereo based FAD uses calibrated non–zero torsional eye positions to warp the disparity surface to simplify the process of detecting structures near the ground plane [9].

The ARK robot navigates in its environment without help from a human operator and with no engineering of the environment through the addition of radio beacons or magnetic strips beneath the floors. Also, modification of the environment to include unique and easily identifiable beacons is also not permitted. The robot uses naturally occurring objects as landmarks. The robot relies on vision as its main sensor for global navigation, using a map with permanent structures in the environment (walls, pillars) to plan its path. While executing the planned path, the robot searches the environment for known landmarks. Positions and salient descriptions of the landmarks are known in advance and are stored in the map. The robot uses the relative position of the detected landmark to update its position. The robot's visual tasks include detection of landmarks and searching for known objects. The robot avoids any objects in its path by using the reactive part of its control system. These objects could be stationary or moving, and do not have to be a part of the internal representation.

In this paper we describe some recent research aspects of the project. In particular we concentrate on environmental path planning, the reactive control system, colour based detection of objects and 3D scene segmentation using the combined visual / range sensor.

2. Mobile Platform and Sensors

We are building two ARK prototypes: one at the University of Toronto and the other at AECL. ARK–1 (at Toronto) is being jointly constructed by university and industry personnel. We use ARK–1 to test the ideas, sensors and algorithms that will ultimately be included in ARK–2. The computing for ARK–1 is done mainly off–board while that for ARK–2 will be done mainly on–board. Both robots use visual data obtained through active vision processes as a primary source of sensing for the robot. They also use non–visual sensors such as infrared, sonar and laser range–finders. Both ARK robots use the Cybermotion Navmaster platform as their mobile base (see figure 2).

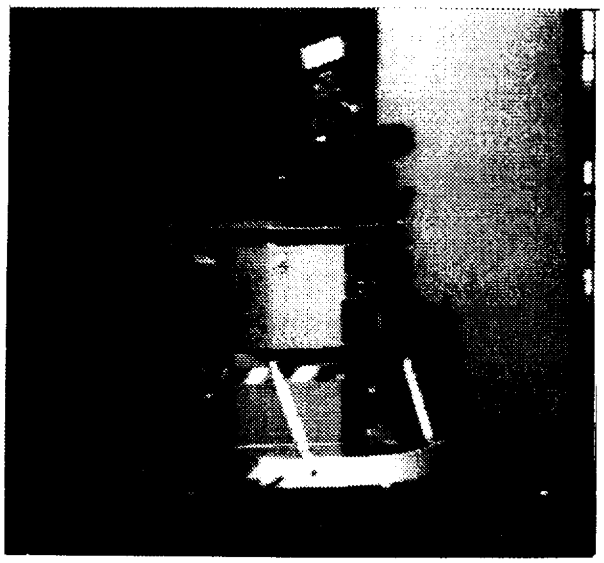


Figure 2. The ARK–1 robot

2.1. Mobile Platform

The main hardware components of the ARK–1 robot are: the Navmaster mobile platform from Cybermotion, the robotic head with sensors and a remote link to a host computer network (figure 2). The platform consists of a base with three wheels and a rotating turret. A bumper, equipped with contact sensors, is mounted to the turret. The turret was originally equipped with six sonars: two of them face forward, two backward and two sideways. Each sonar emits a cone shaped acoustic wave and can detect the reflected wave. The time required by the sound to travel from the robot to an object and back gives a measure of the object distance. We have experimented with using additional sonars mounted on the turret or the bumper to enhance the interpretation of the sonar data.[14]. Multiple return signals were combined in a three dimensional grid in robot coordinates using a Bayesian update rule. Additional readings were obtained by small movements (less than 1 m) of the robot. This approach helped to map more accurately obstacles in front of the robot and to reduce the influence of noisy return signals.

The ARK–1 robot communicates with a network of host computers via the 8–channel remote serial link. The communication between the robot and the host is on the level of processed signals from sensors and commands sent to the robot. The on–board computers collect the data from various sensors, preprocess it and send it via the radio link to the host computer network. The computers in the network analyse this data, and generate commands for individual units of the robot (platform, head, sonar controllers, range–finder). The on board computers perform time critical functions such as emergency stop, positioning the head and moving the platform. The host network of computers consists of a multiprocessor SGI Power Series 4D380 and several Sun SPARC 2 workstations, all running under the Unix operating system.

In ARK–2, most of the computation, such as processing and interpretation of data from various sensors and generation of control commands, will be done on board. The communication link will be primarily used for exchanging messages between the robot and the operator. The on board computer will operate under control of a real time operating system.

2.2. Combined Vision / Range Sensor

We have installed a special sensor (Laser Eye) on the ARK turret. This sensor can provide colour images and range data at distances up to 100 m which are typical for the industrial environment. The Laser Eye is a combined range / video sensor consisting of a camera and a laser range–finder[5]. The range–finder uses the time–of–flight principle and provides a single depth measurement for each orientation of the sensor. Measuring distances to objects in the scene requires pointing the sensor at each of them in turn and reading their depth. The range–finder uses an infra–red laser diode to generate a sequence of optical pulses that are reflected from a target. The time required to travel to and from the target is measured to estimate the distance. The laser is eye safe – this permits its use in the presence of people.

Our robotic head has four degrees of freedom: two extrinsic – head pan and tilt, and two intrinsic – camera zoom and focus (figure 3). The head can tilt in any direction between 65 degrees below and 95 degrees above the horizon and the panning range covers 360 degrees. The head can rotate with speeds exceeding 180 degrees per second. Figure 3 shows the first model of the head with the Laser Eye sensor.

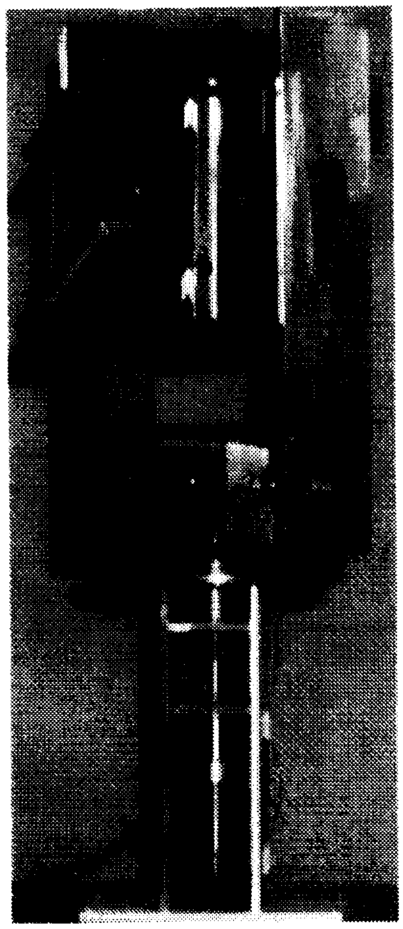


Figure 3. The robotic head with a combined visual & range sensor (Laser Eye)

The range–finder measures distance to an object in the centre of the camera field of view. The co–linearity of the camera optical axis of and that of the range–finder is achieved by using a hot mirror (one that reflects infra–red and transmits visible light) placed in front of the camera lens. The mirror transmits the visible light from the observed scene to the camera with minimum attenuation. The hot mirror reflects the transmitted infra–red beam and sends it in the direction of the optical axis of the camera. The returning pulse is reflected by the hot mirror again and projected on a detector in the range–finder[5]. A single range measurement takes 0.12 – 0.5 second depending on the selected accuracy. The time required to point the head in a new direction depends on the required rotation.

3. Control Architecture

The ARK control system consists of two levels: a high level and a low level reactive system. The high level is responsible for planning robot actions, global path planning, selecting landmarks for sighting and interactions with the user. The low level, reactive component of the control system, uses the on board obstacle avoidance system of the platform to detect obstacles and to navigate around them.

The path planner assumes that the low level reactive control structure will safely execute segments of the plan in the presence of unmodelled or unexpected obstacles. By breaking the path planning process into a GOFAIR (Good Old Fashioned AI and Robotics) task which can be processed using classical AI tools, and a real time reactive process which can be processed using a real time safety critical system implemented as a subsumption architecture, ARK takes advantage of the best of both paradigms.

3.1. Position Estimation and Global Path Planning

The global navigation system uses visual landmarks to update the robot position estimate. A dead reckoning system on the platform measures the distance travelled and provides the current orientation. The positional error introduced by the dead reckoning system accumulates over time and has to be reset by measuring the robot position with respect to landmarks stored in the map. The map is represented as a 2D floor plan that contains permanent objects, semi–permanent objects entered by the user, obstacles detected by the robot and landmarks. Each location in the map is annotated with landmarks that are visible from this location. We use a Kalman filter to update the current position estimate [8].

The global path planning process represents the world as a two dimensional grid. We have experimented with various path planning algorithms such as the shortest path, the minimum cost, and the minimum uncertainty. The shortest path minimizes the distance travelled by the robot and the minimum cost minimizes the number of grid cells visited by the robot. The minimum uncertainty path planner uses the known position of landmarks to choose paths that minimize the expected uncertainty from the start position to the goal. By selecting such a path, the robot may travel a longer distance but its positional error along the path will be much smaller as it can update its position estimate more often.

Figure 4 shows a user interface displaying a map, robot and a planned path. The interface facilitates the creation of a map of the environment, as well as the planning and execution of a path by the real or simulated robot. The high level control system assumes the presence of a low level reactive control system that can execute the path created by the high level.

3.2 Reactive Control

The high level planner communicates with the reactive subsystem through a very simple set of operations that assumes the reactive phase of the planner will operate autonomously and asynchronously; attempting to achieve the current subgoal [12]. The low level control of the robot is based around the subsumption approach described by Brooks [2].

The robot is guided by a set of behaviours that operate in parallel. Each behaviour maps a sensory reading from the robot's environment into an external action of the robot. Conflicting behaviours are arbitrated based on an absolute prioritisation of behaviours. There are three basic behaviours that control the robot: move, avoid, and escape. Avoid watches for an obstacle detected by the front sensing sonar. If an object appears the avoid behaviour stops the robot, and turns it to a new direction so that the robot will not collide with the obstacle. The escape behaviour watches for an obstacle directly in front of the robot, in which case, it causes the robot to back–up and then to turn to a new direction. The escape behaviour helps to get out of certain deadlocks that may occur with the avoid behaviour when the robot gets stuck in a corner. The move behaviour steers the robot towards a precomputed goal position.

Figure 5 shows the planned path and the reactive path executed by the robot as it moves through a doorway. The robot starts in the right top position and moves until it approaches the doorway. At this point, the avoid behaviour is triggered by the edges of the doorway.

4. Using Vision for Navigation

Computer vision plays a major role in the ARK project. The ARK robot uses vision to detect and track landmarks and to search for other known objects. Subsequent surveys and preliminary vision testing have yielded many potential candidates for ARK landmarks in the AECL bay. It is important that these landmarks not only image well but that their occurrence be frequent. Typical landmarks within the AECL laboratory consist of alpha–numeric location signs, fire extinguisher markers, door-

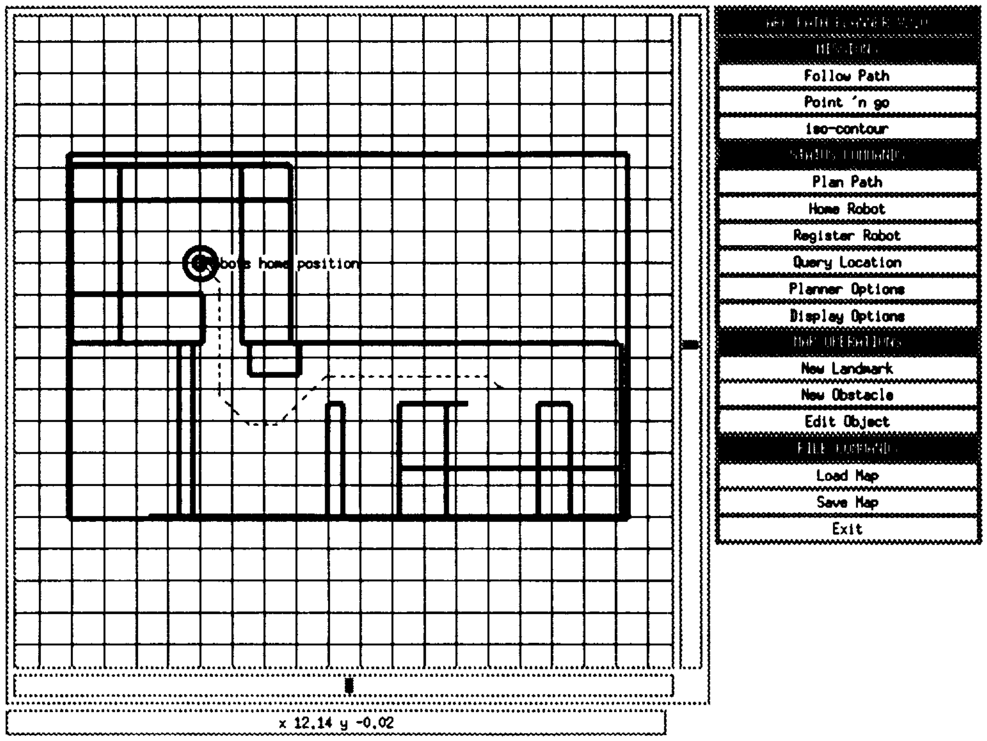


Figure 4. Path planner interface

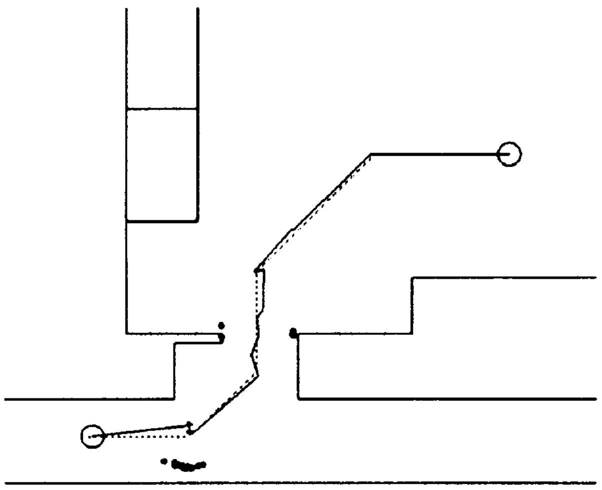


Figure 5. Planned and executed path

ways, overhead lights, and pillars. The only criteria used is that they are distinguishable from the background scene by colour or contrast. These criteria allow the use of both grey level and colour image processing algorithms for landmark identification.

Vision provides important information where to point the range–finder to obtain the most important information. This location depends on the current task, for example, detecting an obstacle or a passage between obstacles. It also depends on the state of a data processing and is driven by an attention model. In two following sections we present results of using vision to detect objects using their colour and to select targets for range measurements.

5. Detecting Landmarks and Objects Using Colour

Visually searching for objects requires scanning the environment or checking expected locations with a camera or even moving a robot. In typical tasks of detecting visual landmarks or searching for a target object, the object itself and its salient characteristic is known in advance. When searching for a landmark the robot can predict where to point the camera as it knows its own approximate location on the map and the coordinates of the landmark. Still, uncertainty of the robot's position requires selecting a wide field of view for the camera. An attention mechanism that selects some "interesting" locations in an image or environment significantly speeds up and simplifies the search. Features such as intensity, colour, high contrast, motion and presence of significant edges are often used to focus attention. Once candidate locations have been selected, each of them is inspected closely to verify presence of the target object.

We use colour to identify possible candidates in an image. The colour classification scheme consists of an off–line training phase and an on–line classification of pixels on a real–time image processor [7]. Colour information is used for pixelwise classification of images and assigning pixels to possible target candidates or background classes. We apply classical methods of pattern recognition for pixel classification. We achieve the real–time performance by creating look up tables (LUTs) during the training phase and fast indexing during the on–line classification.

5.1. Real–time Colour Classification

Classification of every pixel in the image is a computationally expensive task. Modern image processing systems are often equipped with large look up tables that allow for real–time processing of every pixel. Combination of multiple data streams, for example RGB, into one channel enables us to index into the LUT and achieve the real–time performance of an arbitrary (non–linear) conversion. The nature of this conversion is determined by the contents of the LUT. The problem is how to create a LUT that will effectively capture the important variability of the data.

Resolution of the feature space can reach 2^{24} (3 x 8 bit colour bands) for standard colour cameras. Often it is sufficient to operate on smaller arrays. There are hardware limitations as well, for example, the Datacube MV20 advanced processor, used in the project, has a look up table with a maximum of 64 k entries. The contents of look up tables are often determined by manual selection. A more systematic approach uses training by showing examples and manually delineating the objects of interest. Cells in colour space, corresponding to the feature combinations present in the training set, are assigned to appropriate classes. For low resolution of the feature space (200 cells) such a technique is sufficient, as camera noise and blur create dense clusters [13]. For high resolution look up tables containing, for example 64 k cells, this approach is not reliable as insufficient training data creates "holes" in the feature space. Such holes cause misclassification of the data. Various heuristic techniques of filling the space have been used to bridge the gaps [10].

To overcome the problem of the gaps in the LUTs created by limited number of training combinations, we use classical statistical pattern recognition techniques to fill the table. The brute force classification of all possible feature combinations fills the LUT easily.

The training sets consist of images with objects of interest in their natural environment and under different illuminations. Each pixel in the training set is described by its three colour components (RGB or HSI depending on the selected colour space). A clustering programme partitions the three dimensional feature space and creates descriptions for all clusters detected in the training set. After clustering the user assigns individual clusters to classes corresponding to the trained objects and the background. A classification programme uses the description of clusters and their class assignment to process all the pixels in a test image. The test image contains all the feature combinations for a given resolution of the feature space and the resulting LUT will have all its cells filled by this process. Resolution of the LUT is limited by the image processing hardware and in our case the LUT size is equal to 64k (2^{16}). Decomposition of the 24 bit input data into 16 bits can be constant and may always rely on the same algorithm. Alternatively, it may vary depending on the distribution of data in the feature space.

The on–line classification combines the colour components of every pixel into one index to address an entry in the look up table. This entry contains a label corresponding to one of the trained classes.

5.2. Implementation and Results

We have implemented the training phase (clustering and creation of the LUT) on a Unix host. The real–time colour classification is being implemented on the MaxVideo 20 image processing system.

We trained the classifier to detect red and green circular plates similar to the ones displayed on the wall in the scene shown in figure 6. The training set contained multiple plates located in various locations in the scene. The illumination varied between locations. The original pixel

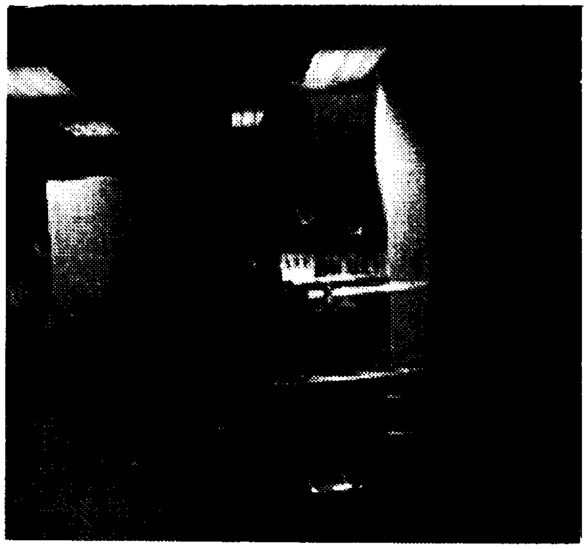


Figure 6. An office scene with coloured objects (luminance is shown only)

values were represented in the RGB space. We used the K–means algorithm to group the data into approximately 20 clusters. The user assigned clusters corresponding to plates to three classes: red, green and the background. This technique is described in detail in [7].

Figure 7 shows the results of pixelwise classification,

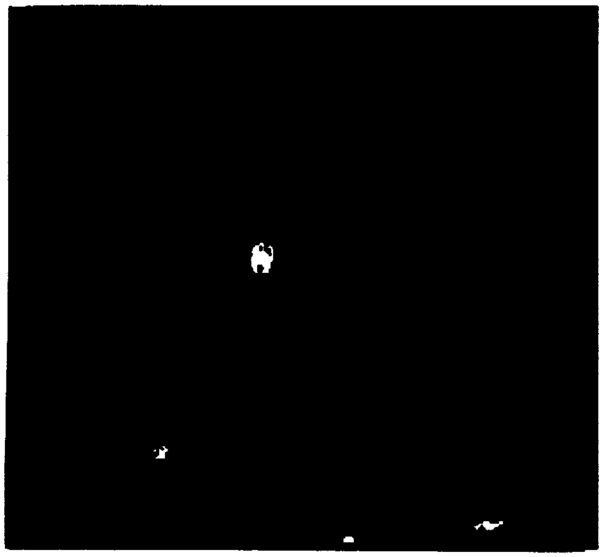


Figure 7. real–time colour detection and reconstruction of object candidates from figure 6

filtering and reconstruction of large blobs representing red and green classes. The results of this processing are not perfect – both red plates have been detected but among the four green candidates only one corresponds to the target object. Also, detection of individual plates is not perfect as regions in the shade or reflecting light are misclassified. Different techniques could be used to decide whether the detected blobs correspond to valid objects or not. At this resolution, however, it might be difficult to decide if the shape deformations are caused by noise, particularly if the sensor is positioned at a difficult viewing angle. It is much better to point the robotic head at every candidate in turn and then acquire and process a new set of images.

Each detected candidate is described by a set of parameters that define its position in the image, size and location of its bounding window. The new orientation of the head is calculated from a kinematic model of the head that includes the pan, tilt and the initial size of the field of view. The new setting for zoom is selected so that the blob of interest is fully included in the new view but dominates the field of view.

6. Using Vision and Range for Navigation

The robotic head with the Laser Eye provides colour images and sparse range measurements at distances up 100 m. With the current version of the head we can obtain sparse range measurements at a rate over 2 Hz. For the real–time operation of the robot it is important to minimize the number of measurements. We use image data to plan where to point the range–finder [4, 5].

6.1. Region Based Image Representation

We assume that nearly all significant depth discontinuities in the scene coincide with the boundaries of detected regions. This assumption requires that the initial segmentation creates an over– rather than under–segmented representation of the image. The under–segmentation can cause potential problems as it requires additional depth measurements to split the region along a depth discontinuity. The size of the regions should not be too small as it is difficult to obtain reliable distance measurements for small regions due to the finite size of the laser spot and accuracy of the robotic head.

The initial segmentation creates an image tessellated into primary regions of homogeneous image properties (intensity, colour, etc.). The segmentation method adopted for the project consists of smoothing, morphological edge detection and the watershed transform. This has been described in detail elsewhere [4]. Large numbers of closed regions of similar image properties are created as a result.

In the image of AECL bay, shown in the figure 1, depth varies from approximately 3 m to 100 m. Figure 8 shows regions detected in figure 1 by the segmentation algorithm. A range map corresponding to this scene can be

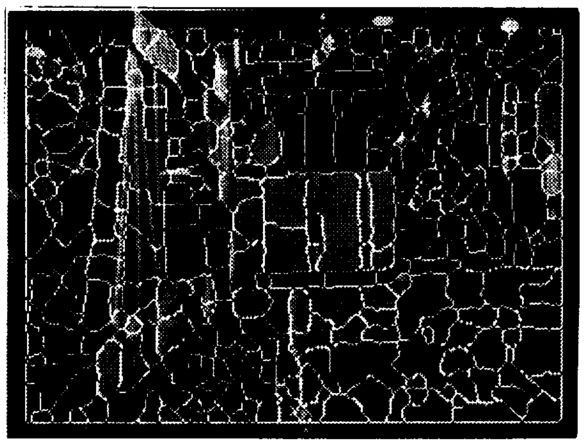


Figure 8. Image from figure 1 segmented into regions

created by selecting target points for each region and pointing the sensor at each of them. The number of targets required for each region depends on the world model and the required robustness. In a simple example, a single range measurement per region yields an approximate range map. Orientation of a planar surface in 3D can be recovered by measuring the distance to at least three points for each region and fitting a plane in Cartesian coordinates. Further processing uses the distances to targets and properties of regions and curves. The result of this processing is a 2 1/2 D representation of the scene.

6.2. Attention Driven Target Selection

In the example shown, the initial segmentation created almost two hundred primary regions. Assuming the simple model with one range measurement per region, creation of the complete range map requires almost 200 range measurements. By applying the above technique we have been able to reduce the number of range measurements required to create the dense range map from 64k samples (sampling every pixel in a 256x256 grid) to a much more manageable number of 200 to 1000 samples (200 regions x 1...5 targets per region). This has been achieved if the initial over–segmentation of the image identified intensity discontinuities and that they account for nearly all the depth discontinuities. For the mobile robot, operating in real–time, this may still be too slow. If we look at the intensity image ourselves, it seems that a few range measurements, taken at the "right" orientations, could provide the essential information essential for a specific task. We decided to look to models of human attention for inspiration.

The attention scheme, used here, depends on three components [6]:

i. *a priori* information,

ii. selection of salient features,

iii. a given task and previous results of attentive processing.

The *a priori* information is encoded as a function biased to look at specific parts of the image. This function represents preferred behaviour (directional sensitivity) of the system, for example, data in the centre or below the horizon might be more important than at the periphery of the camera image.

Representing the segmented image data as a graph allows easy access to underlying regions and boundaries in the graph and for access to adjacent ones. The regions are described by features such as intensity, colour, texture descriptors, and their size and shape. The boundaries between adjacent regions are described by their size, shape, orientation and contrast between regions on both sides. Detection of winners, in the Winner Take All scheme [3], uses a combination of these features and is biased by the specific task performed by the robot.

For example, looking for a passage might involve searching for a dark region in the image. Depth discontinuities are likely to occur at boundaries between contrasting regions. If the task is to provide a qualitative range map, then selecting large regions first will enable faster coverage of the image by range data. Results of previous range measurements can influence the selection of the next target. This selection is task dependent. For example, when searching for an obstacle, if a depth discontinuity is detected, then the next ranging operations should concentrate on recovering the full extent of the closer object and not the distant one. If such a discontinuity is detected while searching for a passage then the successive ranging operations should concentrate on objects further away – the opposite strategy.

Figure 9 shows the attended receptive fields and the path of 10 saccadic movements between regions of high intensity. The initial bias is uniform and contributions from all

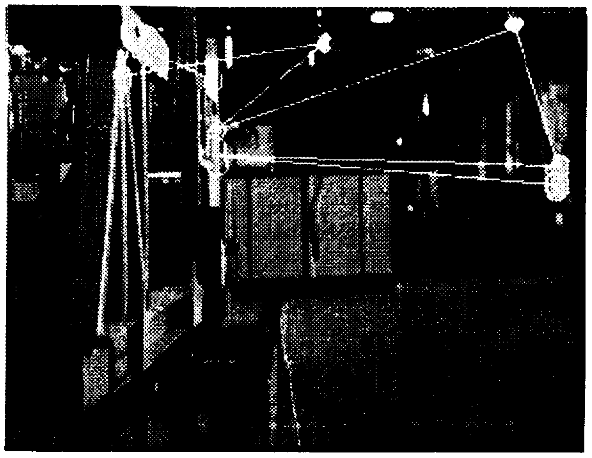


Figure 9. Bright regions selected by a uniformly biased attention model

receptive cells (pixels) are treated equally and, as the result, large bright regions are attended first. Edges of high contrast are likely locations for depth discontinuities. Boundaries between regions now serve as salient features. Pointing the range–finder at a boundary is not practical so two regions on both sides are selected for attention. Figure 10 shows a sequence of saccades between contrasting regions with a bias to the central part of the image. To minimise the number of measurements, each region is attended only once even if it is selected by two different boundaries.

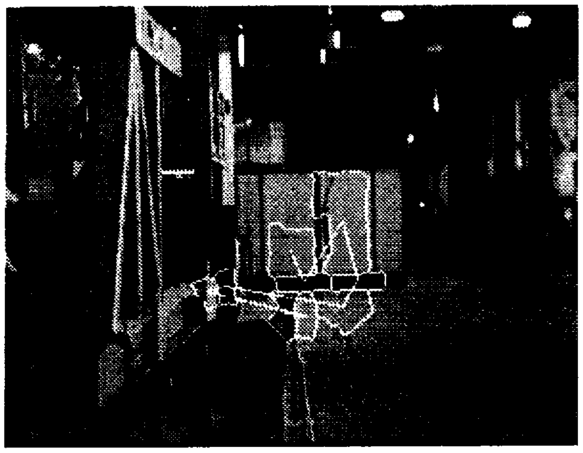


Figure 10. High contrast regions selected by a centrally biased attention model

7. Discussion

The ARK robot relies on its combined vision and range sensor to navigate through the industrial environment. This sensor is unique as it operates at large distances that are typical for the industrial setting. Such distances are not covered by other available techniques used by mobile robots: stereo and active triangulation. Long distance sensory data allows the robot to detect landmarks, search for objects and possible paths well in advance. Early detection of such situations allows the robot to modify its trajectory or to change the plan without the need for an exhaustive search of the environment. Our work concentrates now on extending the reactive, subsumption based, control architecture by implementing additional behaviours. At present, we are moving now with our experiments from the university laboratories to large open spaces of the AECL industrial bay.

One of the strengths of the ARK project stems from the close working relationship between the industrial participants and the researchers from the University of Toronto, York University and the National Research Council.

8. Acknowledgements

Funding for this work was provided, in part, by the ARK (Autonomous Robot for a Known environment) Project, which receives its funding from PRECARN Associates Inc., the Department of Industry, Science and Technology Canada, the National Research Council of Canada, Technology Ontario, Ontario Hydro, and Atomic Energy of Canada Limited.

9. References

1. Blais F., Rioux M., Domey J.: "Optical Range Image Acquisition for the Navigation of a Mobile Robot". Proc. of IEEE Int. Conf. on Robotics and Automation, 1991.
2. Brooks R.: "A Robust Layered Control System for a Mobile Robot". IEEE Trans. on Robotics and Automation, 2(1), 1986, pp. 14 – 23.
3. Culhane SM, Tsotsos JK: "An Attentional Prototype for Early Vision". ECCV–92, pp. 551 – 560.
4. Jasiobedzki P.: "Active Image Segmentation using a Camera and a Range–finder". Applications of Artificial Intelligence XI: Machine Vision & Robotics. Orlando, Florida, April 1993, p. 92 – 99.
5. Jasiobedzki P., Jenkin M., Milios E., Down B., Tsotsos J., Campbell T.: "Laser Eye – a new 3D sensor for active vision". Intelligent Robotics and Computer Vision: Sensor Fusion VI, Proc of SPIE, vol. 2059, Boston, Sept. 1993, pp. 316 – 321.
6. Jasiobedzki P., Service J.: "Recovering Depth by Saccadic Movements of an Active Rangining System". Conference on Vision and Pattern Recognition, CVPR 94 (submitted).
7. Jasiobedzki P., Down B., Service J. Wu V.: "Active object detection using colour and shape". 8–th Canadian Conference on Computer Vision, Signal and Image Processing, Vision Interface 94, Banff, May 1994 (submitted).
8. Jenkin M., Milios E., Jasiobedzki P., Bains N., Tran K.: "Global Navigation for ARK". Proc. of IEEE/RSJ Intenational Conference on Intelligent Robots and Systems, IROS'93, Yokohama, Japan, July 26–30, 1993, pp. 2165–2171.
9. Jenkin M., Tsotsos J.: "Active Streo Vision and Cyclotorsion." Conference on Vision and Pattern Recognition, CVPR 94 (submitted).
10. Massen R., Volk G.: "Real–time colour classification for preprocessing photogrammetry images". SPIE vol. 1395, Close–Range Photogrammetry Meets Machine Vision, pp. 283 – 290.
11. Nickerson B., Jenkin M., Milios E., Down B., Jasiobedzki P., Tsotsos J., Bains N., Tran K.: "ARK – Autonomous Navigation of a Mobile Robot in a Known Environment." Proc. of International Conference on Intelligent Autonomous Systems: IAS–3, Pittsburgh, PA, February 1993, pp. 288 – 296.
12. Robinson M., Jenkin M.: "Reactive Low Level Control of the ARK".8–th Canadian Conference on Computer Vision, Signal and Image Processing, Vision Interface 94, Banff, May 1994 (submitted).
13. Swain M., Ballard D.: "Color Indexing." IJCV 7:1, pp. 11–32.
14. Wilkes, D., Dudek, G., Jenkin, M., and Milios, E., "Multi–transducer sonar interpretation". IEEE Int. Conf. on Robotics and Automation, Atlanda, GA, 1993, vol. 2, pp. 392 – 397.

4025

AIAA-94-1178-CP

BIOLOGICALLY-INSPIRED HEXAPOD ROBOT DESIGN AND SIMULATION

N94-30530

Kenneth S. Espenschied*
Roger D. Quinn†
Department of Mechanical and Aerospace Engineering
Case Western Reserve University
Cleveland, Ohio 44106

p. 8

Abstract

The design and construction of a biologically-inspired hexapod robot is presented. A previously developed simulation is modified to include models of the DC drive motors, the motor driver circuits and their transmissions. The application of this simulation to the design and development of the robot is discussed. The mechanisms thought to be responsible for the leg coordination of the walking stick insect were previously applied to control the straight-line locomotion of a robot. We generalized these rules for a robot walking on a plane. This biologically-inspired control strategy is used to control the robot in simulation. Numerical results show that the general body motion and performance of the simulated robot is similar to that of the robot based on our preliminary experimental results.

I. Introduction

This work is part of an interdisciplinary project which aims to develop practical and robust robot control strategies by using principles extracted from neurobiology. In particular, the problem of hexapod robot locomotion is being addressed, and the primary sources of neurobiological data are the American cockroach, the walking stick insect and the locust.[1-4] A simulation was created to aid in the development of a hexapod robot and its controller because of the relative ease of changing parameters and collecting data.[5,6] We have been building robots for the purpose of further developing, testing, and demonstrating these controllers.

Walking robots have been of interest throughout the history of robotics, including numerous examples with one, two, four and six legs.[7-16] Hexapods are particularly common because they can reposition half of their legs while supporting the body in a statically stable fashion with the other half. With six legs, however, many actuators are required and weight becomes a major design concern. Thus, some method of simplifying the locomotion is often applied, such as the use of pantograph mechanisms which decouple the horizontal and vertical motion.[15,17]

Despite steady progress in the field of robotics, today's walking robots have limited locomotion capabilities compared to insects, which execute this complex task with remarkable skill and robustness. Researchers are making use of biological principles to design robots and their controllers. For example, Raibert has constructed a variety of successful hopping robots controlled based on the principle of the inverse pendulum as in human running.[7,8]

From neurobiology, it is known that there is a close link between the nervous system and the physiology of any animal. In attempting to create a system which achieves successful locomotion by incorporating strategies from the insect world, it may be desirable to start with an insect-like robot. Hence, there is an interest in building biologically-inspired robots and exploiting the synergies found in insects between their mechanics and their control systems. For example, Donner employed a biologically-inspired approach for gait generation in a hexapod robot.[18] Brooks and Ferrell have built small hexapod robots and controlled them using finite state algorithms.[16,19]

Previously, a small hexapod was built and its straight-line locomotion on a flat surface was controlled using a biologically-inspired neural network.[20] The purpose of the robot was to test the controller which was previously developed and demonstrated using a kinematic simulation.[21] This neural network was shown to be robust to the severing of any central or sensory connection.[22] It produced a continuum of statically stable insect-like gaits as a single scalar input governing the speed of the robot was varied.[20] Three mechanisms thought to be responsible for coordination in the walking stick insect were applied to the same locomotion task.[23]

The robot discussed in this paper is more insect-like than the previous robot in terms of leg configuration and degrees of freedom. It is designed to be capable of turning, walking on a rough terrain and walking quickly which requires careful consideration of power and weight. Animal muscle has a high power to weight ratio and controllability that is difficult to reproduce with present technology. The power to weight ratio of DC motors is much less than that of insect muscle. Despite this, DC motors are typically used in robotics because of their controllability.

Every item on a legged robot contributes to the total weight that its legs must lift. It is typical for one leg to support half of the body weight, and in this case, an individual motor may have to support this entire load. A motor which is lightly loaded in one configuration may be heavily loaded in a different configuration, thus, for a highly mobile robot, whose legs may undergo many different configurations, many of the motors must be equally powerful.

In this paper, a previous simulation is reviewed which was developed to assist in the design of the robot, and in particular to help choose appropriate motors and transmissions.[5,6] Next, the design and construction of the robot are discussed. Then, modifications to the previous simulation are introduced to more accurately model the dynamics of the robot. A biologically-inspired controller based on the mechanisms which coordinate the legs of the stick insect is then reviewed. Next, this controller is modified and generalized for the control of the robot walking on a plane. Numerical results demonstrate the locomotion of the simulated hexapod using this controller. The general body motion and performance of the simulated robot are similar to that of the robot based on our preliminary experimental results.

* Research Associate
† Associate Professor, Member AIAA

II. Review of A Simplified Dynamic Model of a Hexapod Robot

Lin and Quinn developed equations which describe the motion of an insect-like walking robot.[5,6] The robot was modeled as having a central body and six legs, each leg having two segments and three revolute degrees of freedom, two where the leg joins the body (hip) and one connecting the two segments (knee). They formulated a simplified dynamic model based on the assumption that the inertia of each leg is much less than the inertia of the central body. This is the case for most insects (for example, all six legs account for approximately 12% of the total mass of a cockroach).

The assumption that the inertia of the leg is much smaller than the inertia of the central body leads to the following conclusions:
(i) Each leg which is in its power stroke (stance) may be treated as if it is in static equilibrium and kinematic equations govern its motion.
(ii) The reactions acting on the body at the hip joint of a leg which is in its recovery stroke (swing) are much less than the reactions at the hip joint of a leg in stance and, therefore, can be neglected.
Hence, the forces and moments at the hips acting on the central body are assumed to be due to the stance legs only. Also, given the joint torques, these forces and moments can be determined approximately based on static equilibrium using the Jacobian matrix of each stance leg.

The central body is treated as rigid with six degrees of freedom. Each stance leg is treated as a manipulator pivoted at the ground contact point with the body treated as its end-effector. On the other hand, a leg in the return stroke is treated as a manipulator with a moving base (the hip). Hence, the equations of motion are decoupled into dynamic equations for each leg which is in the recovery phase, and kinematic equations to represent each leg which is in the stance phase. In comparison with the full dynamic model, the number of equations are the same, but, in the simplified model, the equations are decoupled into a set of less complex systems. Because the equations are decoupled, the leg masses are included in the swinging leg equations as well as in the mass of the body. The leg masses are counted as point masses at their respective hip joints, thus the central body mass is set to the mass of the entire robot. This assumption is justified because the motors, which comprise most of the mass, are located near the hip on the robot described in the next section.

During each time step the simulation is set up as an initial value problem, and given the joint torques, the Newton-Euler equations governing the motion of the central body are integrated to determine the state of the body at the next time step. Then, the equations governing the motion of each leg are integrated to determine its state at that time step. If a leg in its stance phase is found to be in tension, it is switched to the recovery phase. Alternately, when the foot of a swinging leg is found to contact the ground, that leg is switched to the stance phase.

Note that, because the inertia of a stance leg is neglected, the constraint force caused by the ground acting on the foot and the joint forces at the knee joint and at the hip joint are equivalent. Hence, the ground reactions at the foot can be determined from knowledge of the joint torques and will not be unknowns in the simulation problem.

In the simulation, the joint torques and the ground reactions are unknown and are to be determined for a particular walking gait and corresponding joint motions. In general, given a dynamic model of a walking system, when more than one foot is in contact with the ground, a closed kinematic chain is formed and there are an infinite number of solutions to the problem. Pfeiffer et al. used an optimization technique to choose a particular set of feedforward control joint torques.[24] On the other hand, Quinn and Lin used a feedback control strategy to determine the required joint torques to cause the joints to follow the desired joint motions. Both of these strategies have a basis in biology. Lin and Quinn used collocated, proportional-derivative (PD) feedback control which effectively provided active springs and dampers at the joints. The active stiffness and damping gains were chosen to be proportional to the inertia of the link they control. At each time step, the joint torques were determined as proportional to the error between the actual joint motion and the desired joint motion. The ground reactions were then determined using the simplified dynamic model and the equations of motion were integrated as discussed above.

Simulations were performed in which the robot was desired to walk at a constant speed along a straight-line along a smooth horizontal surface. The desired motions of the simulated robot's joints were determined based on metachronal (rear-to-front stepping sequence) insect-like walking gaits. The results showed that each pair of legs displayed a unique insect-like ground reaction force pattern.

III. Design and Construction of a Hexapod Robot

The robot and controller system consists of a personal computer, 18 motor controller circuits contained in a motor controller box, and the robot itself. The computer is connected to the motor controller box with a digital bus, which in turn is connected to the robot by an electrical tether.

The robot, shown in Fig. 1, has a mass of about 5 kg, and is about 50 cm long, 30cm high, and 36cm wide with its legs retracted. The length of an extended leg is about 50cm, and the foot-to-foot distance of opposite, extended legs is about 1.1m. Each leg has three segments, a coxa, a femur, and a tibia, as they are referred to in the insect. The coxa is connected to the body via a revolute joint with its axis perpendicular to the plane of the body (joint 1). The femur attaches to the coxa with a revolute joint with its axis parallel to the body plane (joint 2). Also, the revolute joint connecting the femur and tibia is parallel to the plane of the body (joint 3). Thus, there are three active (motor-driven) joints per leg. In addition, the tibia has a spring loaded linear bearing so that it may compress passively in the axial

Fig. 1. Photograph of the robot.

direction, thus adding a fourth, passive degree of freedom to each leg. The purpose of this degree-of-freedom is to mechanically store energy to augment the actuators, and to reduce impact forces which are generated when a foot contacts the ground. R. McN. Alexander emphasizes the importance of elastic elements in the locomotion of animals, and encourages their application in robotics.[25] We have attempted to incorporate springs in our robot to gain some of the advantages enjoyed by animals.

The robot is constructed mostly of aircraft plywood and balsa wood to minimize mass and inertia. The femurs, which are mostly balsa, are coated with mylar to increase surface toughness. The long, slender section of the tibia is aluminum tube with a rubber tip for a foot. Joint components are mostly aluminum because they are subjected to relatively high stresses. However, the axles for the hip's vertical axis are stainless steel. The attachment between this axle and the body is reinforced with carbon and kevlar fibers. All the joints are supported by ball bearings.

Each of the 18 active joints is driven with a 6 Watt DC motor with an attached planetary transmission. The motors are located near the hip to reduce the inertia of the leg. Joint positions are sensed with potentiometers, and the axial load in each tibia is sensed by a pair of semiconductor strain gages.

To supply an input to the motor, there are digital circuits which make use of pulse-width modulation to control the motor output. The motor controller circuit contains an EPROM so that the control law may be easily modified. Each circuit contains two analog to digital converters. One of these directly converts an analog signal, and this is used for the position feedback. The other one is coupled to a 10x gain to amplify the input voltage before it is converted to digital. This channel was designed for use with the semiconductor strain gages measuring the axial force in the tibia. Also, the joint torque may be estimated by monitoring the output of the motor controller circuits.

IV. Modifications to the Previous Simulation

The net transmission efficiency under the typical operating conditions of the robot was measured to be about 40%. This relatively poor performance is due to the large torques that they transmit to lift the body. Clearly the transmission efficiency plays a major role in the system, contributing to large power losses and reducing backdrivability. Therefore, an adequate simulation of the robot must include a transmission model which reflects this.

Transmission efficiency is related to the load dependent, Coulomb frictional force that results as gear teeth slide upon one another. In developing a transmission model of this phenomenon the difficult problem of modeling a statically indeterminate system is encountered. For example, in the simplest model that includes transmission efficiency, the motor output is multiplied by the efficiency when the motor is doing positive work (driving the joint) and divided by the efficiency when the motor is doing negative work (being backdriven by the joint). In this model a discontinuity occurs when the motor speed changes direction. In fact, the joint torque suddenly changes by a factor of about 5 with 40% transmission efficiency when the speed changes sign. Thus, there is a great potential for instability in this most simple model because of this discontinuity in torque.

Because of the complexity of implementing a truly rigorous transmission model, the simplified model shown in Fig. 2 was developed to represent the frictional characteristics of the transmission. The

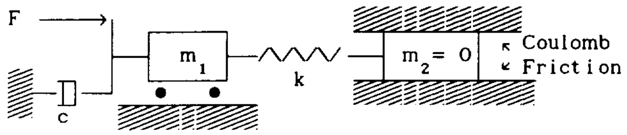


Figure 2. Schematic of motor and transmission model. m_1 represents the inertia of the joint. F is the motor torque. c is a viscous damping constant measured from the motor torque/speed characteristics and k is a stiffness constant. The block on the right is modeled with no inertia and slides on a rough surface subject to Coulomb friction. The maximum magnitude of the Coulomb friction is a function of the motor torque.

purpose of this model is to smooth the above noted discontinuity yet maintain simplicity to permit a straightforward implementation. To account for the torque loss due to transmission inefficiency, a massless auxiliary body was envisioned as added to each joint. This body is coupled to the motion of the joint via a stiff spring. Since the body is massless, the force in the spring is determined only by the frictional force between the body and ground (the stationary side of the joint). The maximum frictional force is limited by the torque output and direction of motion of the motor. Depending upon the sign of the work performed by the motor, the transmission output is described as follows:

$$\tau_{out} = \tau_{mot} + \tau_{loss} \tag{1}$$

where, when the motor is doing positive work, the torque loss is

$$\tau_{loss} = \tau_{mot}(e - 1) \tag{2}$$

and when the motor is doing negative work, the torque loss is

$$\tau_{loss} = \tau_{mot}\left[\frac{1}{e} - 1\right] \tag{3}$$

where τ_{mot}, e and τ_{loss} are the motor torque (the output of the motor multiplied by the transmission ratio), transmission efficiency and torque loss in the transmission due to inefficiency, respectively.

The magnitude of the torque that the spring can apply to the joint is limited to the magnitude of the frictional loss in the transmission by adjusting the position of the auxiliary body. Care is taken not to change the direction of the spring compression when the body slips, as this also would cause a relatively large discontinuity. When the spring is compressed and the auxiliary body is moving with the joint in one direction, then the inefficiency is being modeled accurately. If the velocity then reverses, the spring will decompress as the joint begins to move in the other direction. Eventually, it stretches, and, when the tension in the spring reaches the limit, the auxiliary body begins to slide and accurately model transmission inefficiency again.

This model of transmission inefficiency works best on joints which undergo relatively large motions instead of joints which have high load and maintain nearly constant position over time. The reason is that the spring may store some energy and actually help the motor when the real frictional force would hinder the motor. This effect is minimized by using a stiff spring. However, as the stiffness approaches infinity, the output torque approaches the discontinuity discussed above and instability is imminent. We can determine which joints are effectively modeled by this method from the joint torque, motor torque, and joint velocity data, and interpret the results accordingly. The model may be more useful on undulating terrain than on perfectly flat terrain because the joints will tend to undergo larger motions in this case.

The inertias of the motor rotors were neglected. The reflected value of the rotor inertia is about 40% less than the inertia of the lightest leg segment, the tibia. The loads on this joint when the leg is in the air are very low, and are not of considerable interest.

New graphical output was added to the simulation, along with new code to play back the graphical data files in real time. The previous simulation contained graphic capability, but it was not compatible with the present machine that is running the simulation. The graphical output is of great value in quickly evaluating whether the simulation output is realistic or not, and how natural it appears.

V. Review of Previous Locomotion Controller

As a first step at using a biologically-inspired controller for the locomotion of the simulated hexapod, a generalization of a previous biologically-inspired controller was used. Before describing the modifications, we will first review the operation of the previous controller.[23]

Cruse reviewed three of the mechanisms thought to be responsible for the leg coordination of the stick insect.[3] Dean further describes these mechanisms and shows excellent results for generating insect-like gaits for straight-line locomotion in kinematic simulations.[26,27] In this model of coordination, the insect leg moves between two positions, the Posterior Extreme Position (PEP), and the Anterior Extreme Position (AEP), which are both scalars measured in the body reference frame, where positive is defined as forward. When the leg supports the body and propels the body forward, the foot approaches the PEP. When it reaches the PEP, the foot lifts and moves forward toward the AEP. When it reaches the AEP, the foot is planted and the leg begins to propel the body again. The coordinating influences shift the PEP and AEP from their intrinsic positions, iPEP and iAEP, respectively, and thus phase-shift the stepping cycle of the legs to coordinate them.

Three of these mechanisms were previously applied to the task of straight-line locomotion on a flat surface for a twelve degree of freedom hexapod robot.[23] In this implementation, the coordination mechanisms used only effect the PEP. The mechanisms work to adjust the PEP's in the following way:

1. Each leg produces mechanism outputs unique to that leg. Three mechanisms were used, so there are three mechanism outputs for each leg. The mechanism outputs are plotted in the top three graphs of Fig. 3. These outputs are a function of time and the foot position. The foot position is shown in the lower graph of the same figure.

2. An influence is a dedicated channel through which one mechanism of one particular leg (sending leg) can affect the PEP of another leg (receiving leg). Note that the terms "sending leg" and "receiving leg" are relative only to the influence being discussed. Each influence consists of a weight times the output of the specified mechanism of the sending leg. There is a total of 26 influences in our implementation, all of which have positive weights. Figure 4 illustrates these influences. Each arrow is an individual influence, and the number in the arrow indicates the mechanism that the influence weight multiplies.

3. For each leg, the PEP is adjusted from the iPEP position by an amount equal to the sum of all influences converging on that leg. Notice in Fig. 3 that the position of the foot decreases until it intersects the PEP trace, then it begins to increase. Note, however, that the PEP is adjusted based on influences from mechanism outputs of other legs, not from the mechanism outputs shown in the same figure. The AEP, which is not shown, is a constant, and that is why the position trace always peaks at the same level.

The result of applying this control strategy to the previous robot was a continuous range of statically-stable insect-like gaits, from the slow, metachronal wave (back-middle-front stepping sequence) to the relatively fast tripod gait (middle leg on one

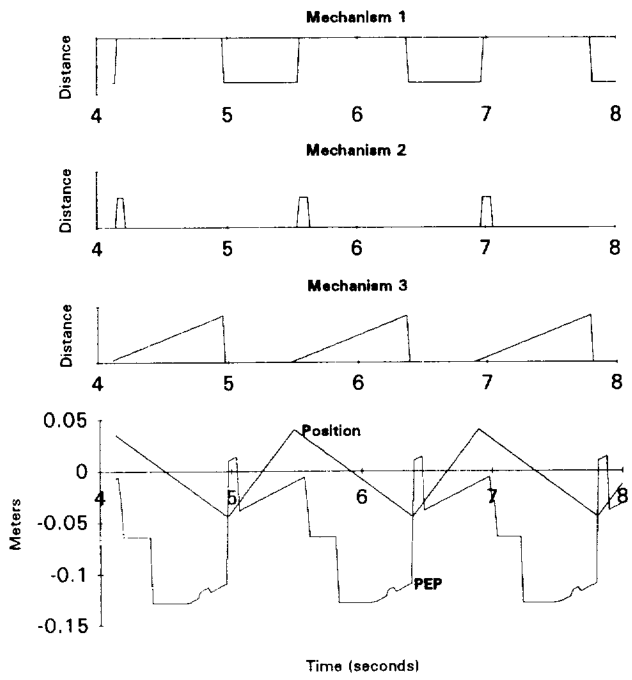


Fig. 3. Leg coordination mechanisms.

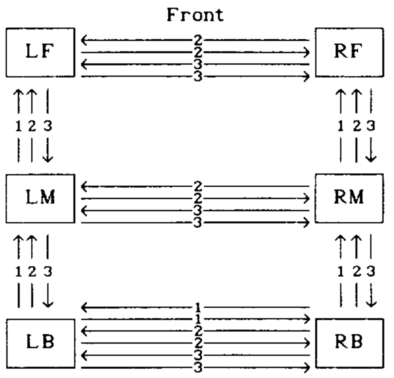


Figure 4. Influences. Each box indicates a leg. L, R, F, M, and B, denote left, right, front, middle, and back, respectively. Each influence is shown by an arrow. The number in the arrow indicates the mechanism to which the influence is proportional.

side of the body steps in unison with the front and back legs on the other side, while each left leg steps in antiphase with the corresponding right leg). There was a single scalar input governing the speed of locomotion, but the resulting gait was produced by the dynamic interaction within the controller and was not pre-programmed. The controller was found to be robust in the sense that it was insensitive to changes in most parameters.

VI. Modifications to the Controller

The new strategy generalizes these rules to locomotion on a plane. The inputs to the controller are forward body velocity, lateral body velocity, and angular rate of body rotation about the vertical axis (yaw rate). The modified controller generates the same range of gaits for forward locomotion, but with the additional ability to "crab" laterally and yaw.

These rules were generalized with the creation of a 1-dimensional variable which is a measurement of the displacement of the current desired foot position from the center of the leg's workspace (home position), in the direction opposite the current foot motion relative to the body. This distance is computed by

$$x = - \frac{\underset{\sim}{x}_{fh} \circ \underset{\sim}{v}_{d}}{\left| \underset{\sim}{v}_{d} \right|} \tag{4}$$

where $\underset{\sim}{x}_{fh}$ is the vector from the home position to the current desired foot position, and $\underset{\sim}{v}_{d}$ is the current desired velocity of the foot relative to the body. The variable x corresponds to the position trace in the lower graph of Fig. 3, and is used to compute new mechanism outputs for each leg, then compared to the PEP and the AEP to determine whether the leg should change states (from power to return stroke or vice-versa).

When the leg is in the power stroke, the desired velocity $\underset{\sim}{v}_{d}$ is computed at each time step. During the return stroke, however, $\underset{\sim}{v}_{d}$ is not calculated. When the leg transitions from power to return stroke, a desired velocity $\underset{\sim}{v}_{dup}$ is computed such that the leg will remain up for a fixed time, and during this time the desired position will move from its present location to where a vector in the direction of $-\underset{\sim}{v}_{d}$ starting at the home position would intersect a circle of radius AEP centered about the home position. Thus, if the desired body motion reverses while a leg is in the return stroke, then it continues its present course until it switches to power stroke, at which time it may begin a new return stroke in the appropriate direction. This approach simplifies the return stroke.

The desired velocity $\underset{\sim}{v}_{d}$ of the foot relative to the body is computed from the desired forward, lateral, and yaw rates of the body in combination with the current desired foot position. Thus, the feet can each have a different desired foot velocity.

The desired vertical coordinate of the foot relative to the body is adjusted based on whether the leg is in the return or power stroke. If the leg is in the return stroke, the desired vertical component is incremented a fixed amount per time step until it reaches the desired maximum, and if the leg is in the power stroke, the desired vertical component is decremented until it reaches the desired minimum.

VII. Numerical Results

The masses, inertias and link length parameters in the simulation were set to correspond to those of the robot. By experimentation we approximated the effective stiffnesses of the robot's joints. For the simulation, we chose the gains for the proportional controller so that the effective stiffnesses of the joints of the simulated robot closely matched those of the robot.

In the previous dynamic simulation, PD control was used.[5,6] The motor model, however, includes viscous damping due to the back emf generated by the motor. Therefore, in the simulation results presented here, we used proportional control only. The motors provide sufficient damping to maintain stability. This was also found to be true for the robot. In the insect, it appears that viscous forces are significant, based on preliminary results from.[28]

The midrange, no-load configuration of the simulated robot is such that the femurs are extended laterally and inclined approximately 45 degrees from the horizontal and the tibias are vertical. Figure 5 shows the graphical output which was added to the previous simulation. The simulated robot is shown as a stick-figure casting a shadow on the plane below it. Note that the simulated robot is under load and walking and, thus, the joints are deflected.

The generalized control scheme described above was interfaced with the modified dynamic model of the robot. The simulated robot successfully walks on a smooth level surface in a continuum of statically stable gaits in response to three inputs: forward velocity, lateral velocity and yaw rate. The general body motion and performance are similar to that of the robot based on our preliminary experimental results. In the simulations, the controller typically causes the simulated robot to settle into a regular gait in just a few steps.

Footfall data illustrating the range of gaits is presented in Fig. 6. Each leg has a trace which is plotted against time, and the trace is only visible when the leg is in the return stroke. These footfall patterns illustrate two features of this controller: The range of gaits that it can produce and the speed with which it settles into these gaits. The top portion of the figure shows the tripod gait and the lower portion shows a slower metachronal wave gait. The middle plot is a medium speed gait. Figure 7 shows the body roll and pitch during the tripod gait shown in Fig. 6.

Because the particular influences chosen were based on forward walking of the stick insect, during sideways or even backwards stepping the gait is still a back-middle-front metachronal wave. In future work we may adjust these influences based on the desired direction of motion. We would like to emphasize that the sideways and backwards gaits are statically

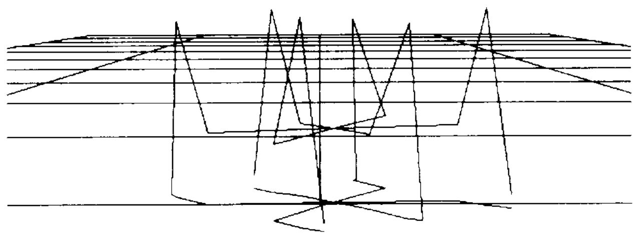


Fig. 5. Simulation environment.

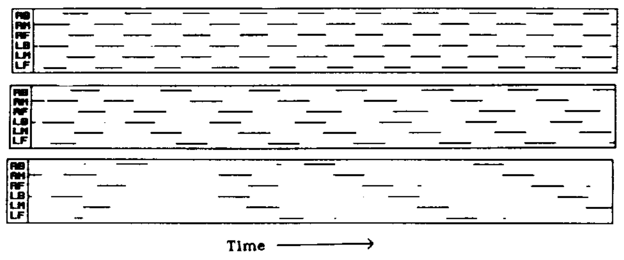


Fig. 6. Stepping patterns for several gaits.

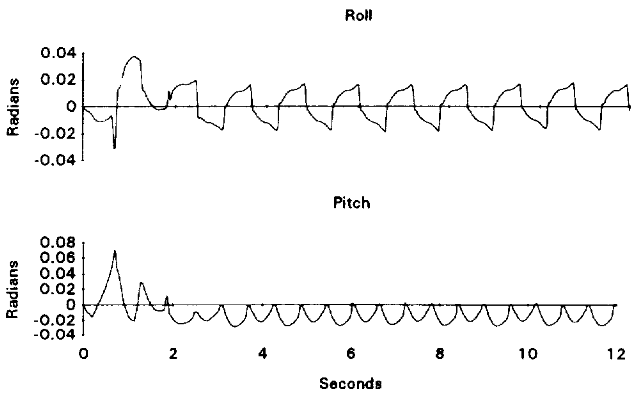


Fig. 7. Roll and Pitch of body during tripod gait.

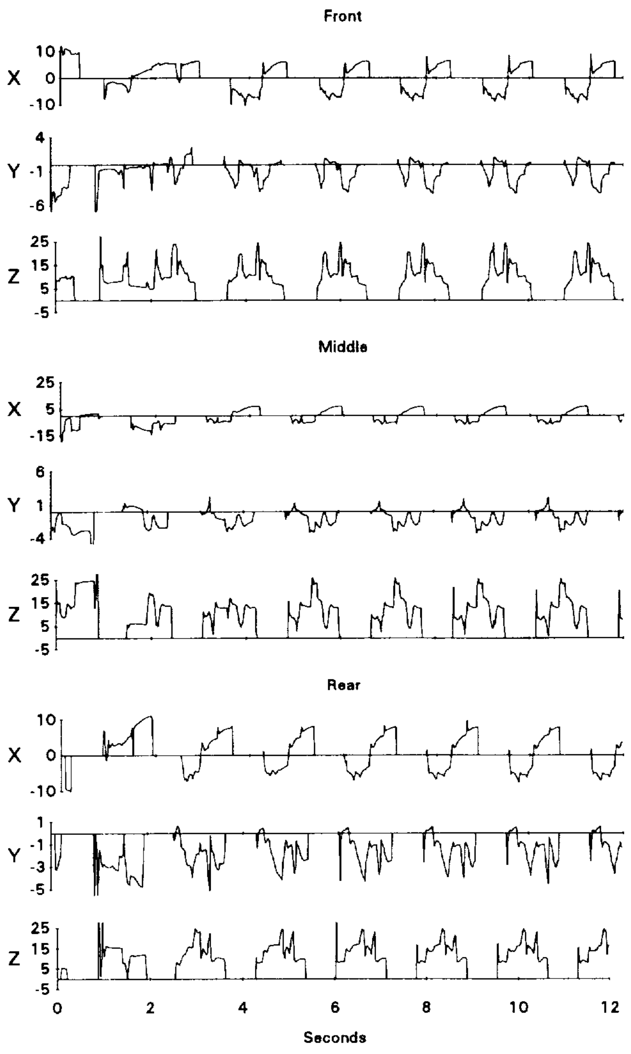


Fig. 8. Ground Reaction forces for left legs.

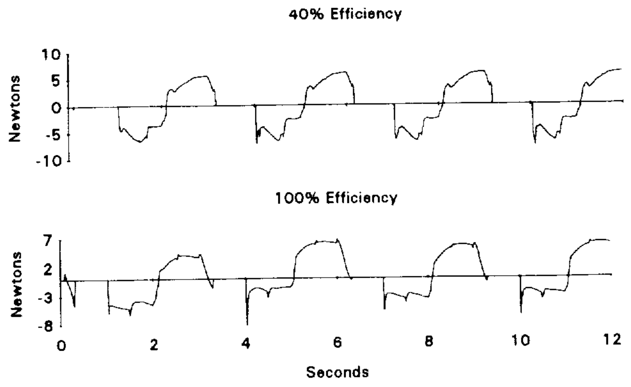


Fig. 9. Effect of transmission efficiency on x ground reactions (LR leg).

stable, but not necessarily insect-like nor optimal for static stability. The controller does sometimes try to lift two adjacent legs when the inputs are changed quickly, but it does adequately maintain static stability when the input is changed gradually, and allows for a wide range of walking behavior.

Figure 8 displays the ground reaction forces for the three left legs while the simulated robot walks in the medium speed gait shown in Fig. 6. In these figures, the X direction is forward, Y points to the left, and Z is upward relative to the body. Note that while the simulated robot is walking at a steady average speed, the front legs tend to decelerate the body, the rear legs tend to accelerate the body, and the middle legs first decelerate then accelerate the body during their respective drive phases. The lateral (Y) forces are directed toward the body for all legs. Similar force patterns have been observed for insect locomotion.[2] The previous simulation, in which PD control was used, also exhibited this insect-like force pattern.[5,6] However, the effect in the X direction was more pronounced than in this modified simulation. Figure 9 shows the ground reaction forces in the X direction for the left rear leg using a transmission efficiency of 40% and 100%. This effect is more pronounced in the 100% efficiency case. We conclude from this that Coulomb friction is responsible for this difference.

Figure 10 shows the position versus time for joint 2 (front to back swing) of the left middle leg, which corresponds to the medium speed gait shown in Fig. 6. The function of the transmission model (see Fig. 2) is illustrated in Fig. 11 which shows motor torque (multiplied by transmission ratio) and total joint torque vs. time for joint 2 (front to back

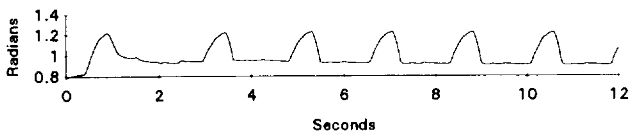


Fig. 10. Joint 2 position vs. time (LM leg).

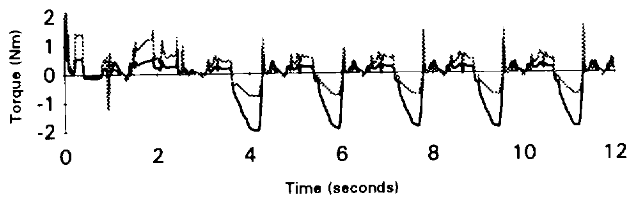


Fig. 11. Motor and joint 1 torque vs. time (LM leg).

swing) for the left middle leg. Note that when the leg is in the recovery stroke the motor is doing positive work and its torque is higher than the joint torque. In the first half of the power stroke, the motor does negative work (slows the body), and in the second half it does positive work (propels the body). Notice that the magnitude of the motor torque is less than the joint torque during the negative work phase (when torque is negative in this case) and greater than the joint torque when the work is positive (positive torque in this case) as one would expect from transmission inefficiency.

VIII. Summary

The design and construction of a small 18 degree of freedom robot is described. The robot is designed to walk on rough terrain. We modified a previous simulation of an 18 degree of freedom hexapod to increase its utility for the task of design and modeling of a hexapod robot. The most significant modifications were to add models of the motor driver circuit, motor, and transmission, including a simplified model of transmission inefficiency. A previously designed biologically-inspired locomotion controller, which originally produced straight-line forward locomotion on a flat surface, was generalized to produce lateral and turning motion. This generalized control scheme was interfaced with the modified dynamic model of the robot. The simulated robot successfully walks on a smooth level surface in a continuum of statically stable gaits in response to three inputs: forward velocity, lateral velocity and yaw rate. The general body motion and performance are similar to that of the robot based on our preliminary experimental results. In the simulations, the controller typically causes the simulated robot to settle into a regular gait in just a few steps. The ground reaction forces generated by the locomotion share significant features with force data on insect locomotion.

Acknowledgements

We would like to thank Harold P. Frisch of the NASA Goddard Space Flight Center for grant NGT-50588 and Dr. Thomas McKenna of the Office of Naval Research for grant N00014-90-J-1545. The Cleveland Advanced Manufacturing Program supplied additional funding through the Center for Automation and Intelligent Systems Research at CWRU.

IX. References

1. Ritzmann, R. E., "The neural organization of the cockroach escape and its role in context dependent orientation," In R. D. Beer, R. E. Ritzmann, and T. McKenna (Eds.), *Biological neural networks in invertebrate neuroethology and robotics*. New York: Academic Press, 1992.
2. Full, R. J., Blickhan, R., and Ting, L. H., "Leg design in hexapedal runners," *Journal of Experimental Biology*, 158, 369-390, 1991.
3. Cruse, H., "What mechanisms coordinate leg movement in walking arthropods?" *Trends in Neural Science*, 13, 15-21, 1990.
4. Pearson, K. G., and Franklin, R., "Characteristics of leg movements and patterns of coordination in locusts walking on rough terrain," *The International Journal of Robotics Research*, 3(2), 101-112, 1984.
5. Lin, N.J., "Dynamics and Control of Open- and Closed-Chained Multibody Systems," Ph.D. Dissertation, Case Western Reserve University, 1992.
6. Quinn, R. D. and Lin, N. J., "Dynamics and Simulation of an Insect-Like Walking Robot," Proceedings of the Ninth VPI&SU Symposium on Dynamics and Control of Large Structures, May 10-12, 1993
7. Raibert, M.H., *Legged Robots that Balance*. Cambridge, MA: The MIT Press, 1986.
8. Hodgins, J.K. and Raibert, M.H., "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics and Automation*, 7, 289-298, 1991.
9. Miura, H. and Shimoyama, I., "Dynamic walk of a biped," *The International Journal of Robotics Research*, 3, 60-74, 1984.
10. Liston, R.A. and Mosher, R.S., "A versatile walking truck," *Proceedings of the 1968 Transportation Engineering Conference*, ASME-NYAS, Washington, D.C, 1968.
11. Raibert, M.H., Chepponis, M., Brown, H.B. Jr., "Running on four legs as though they were one," *IEEE Journal of Robotics and Automation*, RA-2, 70-82, 1986.
12. Hirose, S. and Kunieda, O., "Generalized standard foot trajectory for a quadruped walking vehicle," *The International Journal of Robotics Research*, 10, 3-12, 1991.
13. McGhee, R.B., "Vehicular legged locomotion." In G.N. Saridis (Ed.), *Advances in Automation and Robotics*. JAI Press, 1983.
14. Byrd, J.S. and DeVries, K.R., "A six-legged telerobot for nuclear applications development," *International Journal of Robotics Research*, 9, 43-52, 1990.
15. Song, S. M. and Waldron, K. J., *Machines that walk*. Cambridge, MA: The MIT Press, 1989.
16. Brooks, R.A., "A robot that walks; emergent behaviors from a carefully evolved network," *Neural Computation*, 1, 253-262, 1989.
17. Hirose, S., "A study of design and control of a quadruped walking vehicle," *The International Journal of Robotics Research*, 3, 113-33, 1984.
18. Donner, M.D., *Real Time Control of Walking*. Boston: Birkhäuser, 1987.
19. Ferrell, C., "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators," Ph.D. Dissertation, M.I.T., May 1993.
20. Quinn, R.D. and Espenschied, K.S., "Control of a hexapod robot using a biologically inspired neural network," In R.D. Beer, R.E. Ritzmann, and T. McKenna (Eds.), *Biological Neural Networks in Invertebrate Neuroethology and Robotics* (pp. 365-381). New York: Academic Press, 1992.

21. Beer, R. D., and Chiel, H. J., "Simulations of cockroach locomotion and escape," In R. D. Beer, R. E. Ritzmann, and T. McKenna (Eds.), *Biological neural networks in invertebrate neuroethology and robotics.* New York: Academic Press, 1992.
22. Chiel, H. J., Beer, R. D., Quinn, R. D., and Espenschied, K. S., "Robustness of a distributed neural network controller for locomotion in a hexapod robot," *IEEE Transactions on Robotics and Automation*, 8, 293-303, 1992.
23. Espenschied, K. S., Quinn, R. D., Chiel, H. J., and Beer, R. D., "Leg coordination mechanisms in the Stick Insect Applied to Hexapod Robot Locomotion," *Adaptive Behavior*, 1(4), 455-468, 1993.
24. Pfeiffer, F., Weidemann, H. J., and Danowski, P., "Dynamics of the Walking Stick Insect," *IEEE Control Systems*, pp. 9-13, February 1991.
25. Alexander, R. McN., "Three uses for springs in legged locomotion," *The International Journal of Robotics Research*, 9(2), 53-61, 1990.
26. Dean, J., "A model of leg coordination in the stick insect, *Carausius morosus*; III. Responses to perturbations of normal coordination," *Biological Cybernetics*, 66, 335-343, 1992.
27. Dean, J., "A model of leg coordination in the stick insect, *Carausius morosus*; IV. Comparisons of different forms of coordinating mechanisms," *Biological Cybernetics*, 66, 345-355, 1992.
28. Marx, W. J., Beer, R. D., Nelson, G. M., Quinn, R. D., and Crocker, G. A., "A Biomechanical Model of the Cockroach," presented at the Annual Meeting of the Society for Neuroscience, Washington, D.C., Nov. 7-12, 1993.

6026

AIAA-94-1180-CP

ODYSSEUS AUTONOMOUS WALKING ROBOT: THE LEG/ARM DESIGN

//13

N94- 30531

N.G.Bourbakis, M.Maas, A.Tascillo and C.Vandewinckel

Binghamton University
T.J.Watson School
AAAI Lab
Binghamton, NY 13902

p 8

ABSTRACT

ODYSSEUS is an autonomous walking robot, which makes use of three wheels and three legs for its movements in the free navigation space. More specifically, it makes use of its autonomous wheels to move around in an environment where the surface is smooth and not uneven. However, in the case that there are small height obstacles, stairs, or small height unevenness in the navigation environment, the robot makes use of both wheels and legs to travel efficiently. In this paper we present the detailed hardware design and the simulated behavior of the extended leg/arm part of the robot, since it plays a very significant role in the robot actions (movements, selection of objects, etc.). In particular, the leg/arm consists of three major parts: The first part is a pipe attached to the robot base with a flexible 3-D joint. This pipe has a rotated bar as an extended part, which terminates in a 3-D flexible joint. The second part of the leg/arm is also a pipe similar to the first. The extended bar of the second part ends at a 2-D joint. The last part of the leg/arm is a clip-hand. It is used for selecting several small weight and size objects, and when it is in a "closed" mode, it is used as a supporting part of the robot leg. The entire leg/arm part is controlled and synchronized by a microcontroller (68CH11) attached to the robot base.

Keywords: Autonomous Walking-Wheeled Robots; Robot Design; Robot Leg/Arm;

This work is a part of the ODYSSEUS research project (FRG grant 1992-93).

1. INTRODUCTION

The study of autonomous walking robots (AWR) is a very attractive area of research and human challenge, since AWRs provide a better mobility in terrains with irregularities than wheeled robots. In particular, in buildings with many floors and stairs, with no access to elevators (in case of fire or earthquake), or floors with surfaces of different levels, wheeled robots are almost useless beyond a one-level surface. Moreover, if for some reason there is a blocked corridor, (e.g. because of a low height obstacle dropped accidently), a wheeled robot has to return to look for another open corridor in order to reach the destination point. On the other hand, on floors with no irregularities wheeled robots (so far) move faster than walking robots and the control of their motion is simpler than walking ones.

A variety of AWRs have been designed, constructed or proposed to fulfill either new challenging ideas or application needs [1-8]. In particular, the NOMAD walking robot was constructed by undergraduate students for participation in a walking robots competition [1]. It consists of two triangular platforms, where each platform carries three legs located at the triangle's corners. Nomad walks by rotating around its legs. This design presents difficulties, however, such as instability on uneven terrains. The AMBLER walking robot is under development for the exploration of the planet Mars [2]. It uses six legs, three from each robot side. The robot will move by rotating the legs and follows a direction within an angle of 30 degrees. The robot's size is very large, with each leg having a height of seven meters. It has the ability to step over objects three feet high. This robot project, however, stopped due to the infeasibility of transferring a large and very heavy structure with today's space capabilities. Another walking robot is MRSR [3]. It was developed for the Mars space project and uses two platforms. The square platform holds four legs at its corners, and the triangular platform carries three legs at its own corners. The triangular platform surrounds the square one. It walks by moving triangular platform ahead and when it is stable the square platform follows by rolling on a common rail bar. It is a stable robot with the capability to walk on uneven terrain. A small walking robot with six legs was constructed by Brooks to study the integration of a complex robot machine within a large number of sensory inputs [4]. The robot uses six legs (three on each side) and is about 35 cm long. Each leg is rigid and is attached at a shoulder joint with two degree of rotation freedom, driven by two orthogonally mounted model airplane position controllable servo motors. Due to the small size of this robot it can be used as a tool for the study of microrobotics [9].

Since the walking robot research field is "open" with unsolved problems and new challenging ideas, a new hybrid (wheeled/legged) robot, called ODYSSEUS, is presented [10,11]. It uses a triangular platform on which three autonomous-extended wheels are attached at its corners, while three legs/arms are located at the middle of each triangle side. Note the first version (wheeled) of ODYSSEUS was constructed by accommodating the study and design of distributed sensory input data (sonar, vision) for the extraction and abstract modeling of the navigation space [10,12,13], and other important navigation issues.

In this paper the structural design and the first stage feasibility simulation of the leg/arm of ODYSSEUS are presented. A brief description of the main features of ODYSSEUS is given firstly. The design section includes the detailed description of the leg/arm parts (joints, motors, shoulder, elbow and hand). The functional section includes the operation of each part and the conditions under which these part function. The last section provides a simple simulation of the leg/arm global operation.

2. ODYSSEUS ROBOT

In this section the main structural and operational features of the autonomous mobile robot, called ODYSSEUS are presented briefly.

2.1 STRUCTURAL DESIGN

2.1.1 Triangular Base

The choice of the base configuration was determined by the robot's primary objective of being capable of climbing stairs. Additionally, it was desired that the robot to have accessibility to as large an area as possible. After a careful consideration it was concluded that the triangular base (Figure 1) would best meet our objectives. Unlike a circular base, the triangular base can reach a corner in a room.

Attached to the base are the legs, arms, power supply, navigation system, and control units. Underneath the base, in the center, a battery is attached. The battery is the power supply of the robot. At the robot's base corners, autonomous, programmable legs/wheels are connected to a rail system. Three legs/arms will also be attached appropriately to the middle of the base's sides, Figure 2. On the top of the base are the navigation systems, the main processor and slave processors.

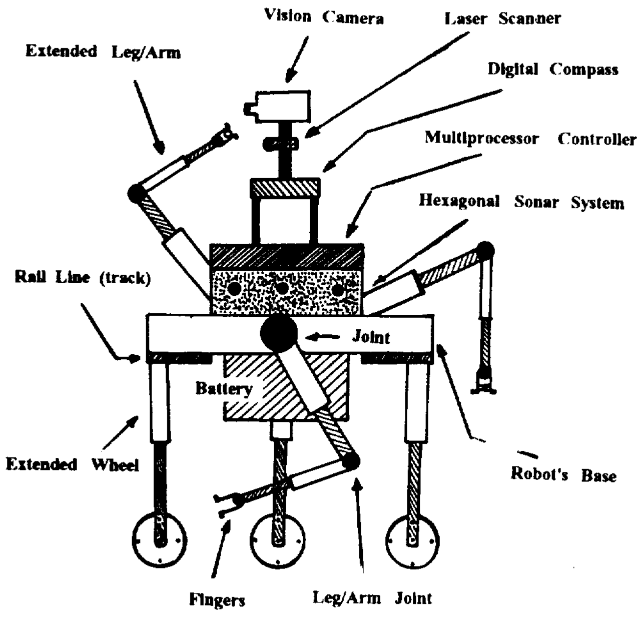


Figure 1: The ODYSSEUS Robot.

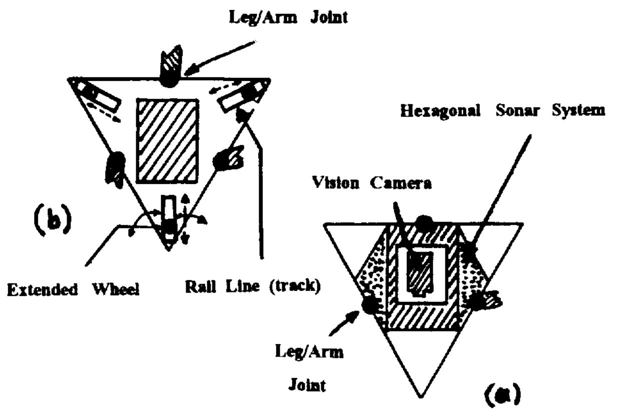


Figure 2: a) Robot's top view. b) Robot's bottom-up view

2.1.2 Navigation System

By using a digital compass the robot has the ability to orient itself in an environment. The compass utilizes three magnetic sensors to produce a digital readout of the robot heading. A laser sensor is used to measure the distance from various objects. The distance from the objects is combined with the view from a camera to provide a three dimensional image of the space [14]. This image assists the robot in its planning strategy.

2.1.3 Extended Autonomous Leg/Wheel

The extended wheel consists of three main parts. The first part, called the basic-pipe, is attached underneath the robot's base to a rail-line. The rail-line starts from the corner of the base and ends at a distance $dr \geq ww + sw$, where "ww" represents the maximum diameter of the wheel and "sw" is a safety factor. Inside the basic-pipe is the second part, called the extended-pipe. This feature allows the leg/wheel to be extended or shortened. At the other end of the extended-pipe, the third part, a wheel, is attached. Four holes in the wheel are used in the calculation of distance traveled and velocity of the robot. The wheel also has the capability of rotating. Determination of rotation angle is calculated by the main processor.

2.1.4 Extended Autonomous Arm/Leg

A detailed description of the leg/arm is provided in this report. It has the capability of grabbing and moving an object. Additionally it also has the capability of assisting the legs when the robot is in the stair climbing routine [11].

2.1.5 Distributed Multi-microprocessor System

Since each robot part has its own associated microprocessor, a multi-microprocessor distribution system is formed. Each microprocessor (in particular a Motorola 68CH11) controls and processes information related to that robot part. A central master microprocessor is used to establish communication with all the other microprocessors. The master microprocessor will synchronize and optimize the operation of the distributed microprocessor system.

Specifically, the main processor will receive processed information from the associated microprocessors, combine the information, and make the appropriate decisions for the next robot action (movements, selection, rotation, synchronization of the wheels, etc.). The master processor shares common memories with each of the associated processors.

3. DESIGN OF THE EXTENDED LEG/ARM

The design specifications for the extended leg/arm are explained in this section.

3.1 Brief Overview

A design of the entire arm is shown in Figure 3. It basically contains six motors: two dual purpose motors (labeled motor one and motor two), three elbow motors, one at each joint, and one motor for the hand. Clearly the elbow motor at the base is more powerful than the motor at the hand. This design also includes two extension units that are used during stair climbing programs. The bearing lock system provide motors 1 and 2 the additional capabilities of extending and twisting the arm.

3.2 Detailed Design of the Extended Leg/Arm

A primary concern in the design of the arm was how to avoid the "Popeye syndrome". If one can recall Popeye's forearms were much larger than his upper arm. In this deign we wanted to minimize the weight at the end of the arm to limit excessive stress on the components. However, the arm must still be able to support at least a third of the robot body weight.

3.2.1. Joint Design and Operation

The elbow joint, shown in Figure 4, consists of a motor inside of a shell. The shell has two parts; an inner shell which is mounted to the motor and an outer shell which is free to rotate about the center line of the motor. On the outside of the elbow motor are two plate mounts which can be used for additional sensory components in future design revisions. The outer shell is attached to one of these plates and the inner shell is attached to the other. When the motor rotates, the rotating gears (which are fixed in the motor) rotate, one in the opposite direction with respect to the other. Both of these gears are connected to a third cylindrical gear called the shell gear. The motor's rotation causes the plate mounts to move in opposite

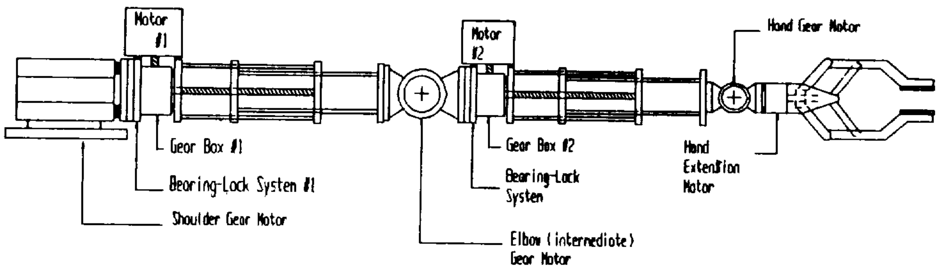


Figure 3: The Leg/Arm design

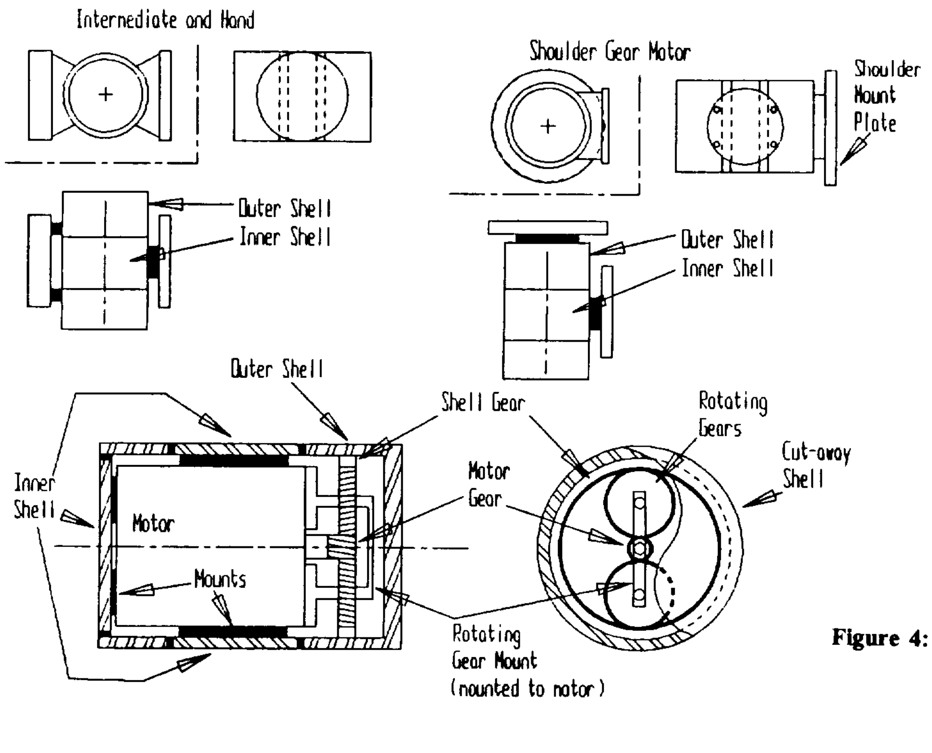


Figure 4: The gear motor designs

rotational directions. A primary concern of this system is the torque developed. This will depend on the motors used and on the gear ratio of the motor gear to the cylindrical shell gear. We will use this type of system for each elbow location (i.e. shoulder, intermediate, and hand joints). Obviously the hand will have the smaller and lighter system compared to the shoulder.

3.2.2. Shaft Design and Operation

This section contains an explanation of the extension units, the bearing lock systems, and motors one and two. The extension units consist of four plates, six support rods and one large threaded shaft. This unit is shown in Figures 5 and 6. Plates 1 and 3 are secured to each other via support rods. The same applies to plates 2 and 4.

The concept of this design is very simple. The threaded shaft rotates while plates 2 (which is also threaded) and 4 ride on this shaft. They extend or contract depending upon the rotation of the threaded shaft. Assuming that the screw will have a right hand thread, this system will extend when the shaft is rotating counter clockwise and will contract when the shaft is rotating clockwise.

Plate 1 is a mount plate that will be mounted on the gear box. The mount plate has three mount holes and one large shaft hole that should be larger than the threaded shaft. Plate four has the same purpose as plate 1. Plates 2 and 3 have the dual role of supplying support and being

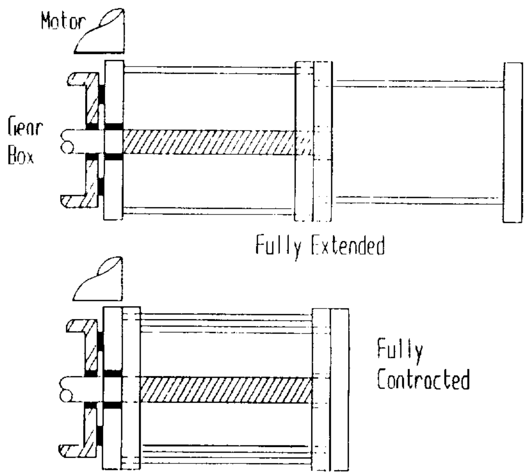


Figure 5: The extension unit

able to move. It is anticipated that all plates and shafts will be constructed from 6061 Aluminum.

By using the isosceles plate and rod design instead of the cylindrical shaft inside a shaft design we minimized the weight of the unit. Our design should supply ample support in all directions. Three support rods, instead of two, are used to assure support when twisting torque is applied to

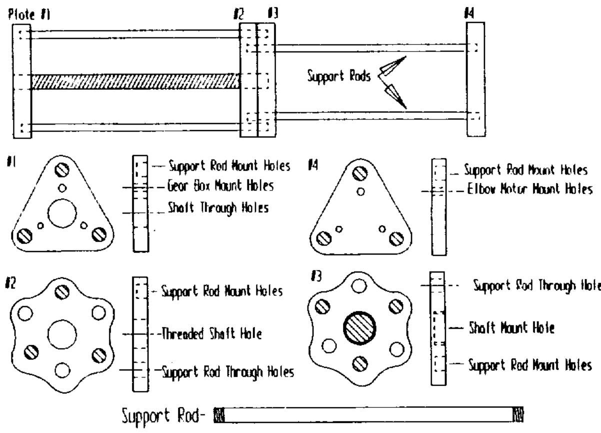


Figure 6: The extension unit detail design

the extension unit. However, we sacrifice speed when we use the screw style extension unit. The reason behind the loss of speed is the steep screw pitch that is required to support the weight.

The bearing lock system is probably the most complex component of the arm. It can be observed in Figure 7. Its purpose is to allow motors one and two to operate to two degrees of motion. One, an extension motion explained earlier in this section, and the second a twisting motion explained later. Using one motor for two purposes simply eliminates the need for another motor and reduces the overall weight of the arm.

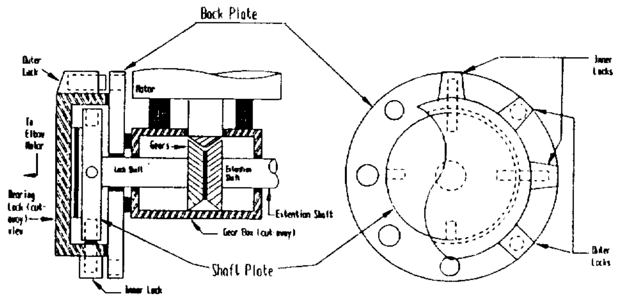


Figure 7: Bearing lock system

The bearing lock system consists of a gearbox which contains one shaft that goes to the extension unit and another very short shaft that becomes part of the lock system. The lock system consists of the two mount plates, six shaft locks, three inner locks, and three outer locks.

The bearing lock that is shown as a cut away will be mounted to the elbow motors at both the base and the intermediate positions. Locking the outer locks will secure the back plate and therefore the motor, gear box, and the extension unit to the elbow motors. When this is done, the inner lock should remain unlocked. This allows the motor to be used for extension purposes. Locking the inner lock and unlocking the outer locks will cause the small lock shaft to be secured to the elbow motors. At this time, when power is supplied to the motor, the motor gear box and extension shaft rotate around the fixed lock shaft. This supplies a twist to the components of the arm that are beyond the bearing lock system. It should be noted that this system does not give freedom of either the extension operation or the twisting operation. The extension units will be in motion at all times. However, this should not hinder the arm's performance significantly since the arm will extend very little with any twist.

In addition the bearing lock system can also be used for locking the extension unit. It is required that the extension unit will be locked without twisting in order to support a large amount of weight. This is accomplished by simply locking both locks.

Several of the bearing lock faults should be noted. Besides the difficulty in manufacturing the system, a performance flaw exists. Each lock needs to locate a hole in order to serve its purpose. Alignment of these locks may become very difficult. Especially if the extension unit is extended all the way and the lock's cannot be aligned with the hole. In this case the extension unit would have to be contracted in order to align the holes. This difficulty can easily be corrected by having a alignment solution program in the arm's microprocessor.

Another problem is with the locks. The ones initially chosen were magnetic locks. However, their locking power is in question. Also, when supporting a large portion of the robot weight, unlocking may not be feasible. In future design revisions a gear lock may be more suitable in alleviating these two problems.

The final component of the arm is the hand. It can be seen in Figure 8. Here again, simplicity is evident. When the driver bolt moves on the threaded shaft, the slotted rods ride on the fixed pins and force the fingers open or closed.

The major problem with the hand is structurally it is the weakest part of the arm yet is still needs to support a large amount of weight. Therefore, a different material (i.e. stainless steel) will be used for the hand.

Grabbing strength of the hand depends on the torque of the motor, the pitch of the screw threads, and grabbing method design. Since the motor of the hand will be the smallest one on the arm, it will not have the same strength as the other motors. A steep thread pitch is needed to assure grabbing strength.

4. FUNCTIONAL DESCRIPTION OF THE EXTENDED LEG/ARM

This section provides a macroscopic view of the arm design as it applies to the two major functions of the arm: grabbing and assisting the robot in climbing stairs.

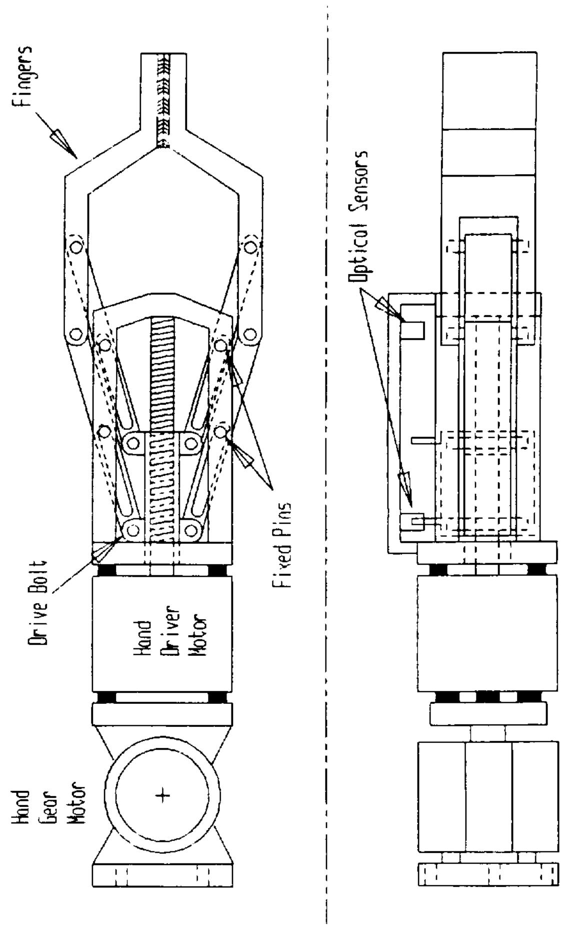


Figure 8: The hand

4.1 Extended Arm/Leg Functional Parameters

It is required, during stair climbing, that the arm design be capable of lifting the Odysseus robot upward approximately one foot. Since there will be three arms on the robot, two of these will be used at any time for the lifting operation. The Odysseus robot is expected to weigh 150 pounds in the worst case. During the lifting operation about 1/3 of the robots weight will be supported by the hind leg. This requires the use of motors one and two during this movement.

During the time that the arm is used to simply grab and lift objects, all components of the arm are utilized during this simple operation. Several limitations exist during the arm grabbing operation. The designed arm can contract to 35 inches and extend to 47 inches when straight. It can grab as wide as your average soda can. Since the shoulder only has one degree of freedom, this limits its grabbing reach. From any one side of the base of the robot the arm has the capability of reaching as far out as the lower portion of the arm allows. This translates into a 27.5 inch reach with a 360 degree swing around the shoulder axis, as seen in Figure 9.

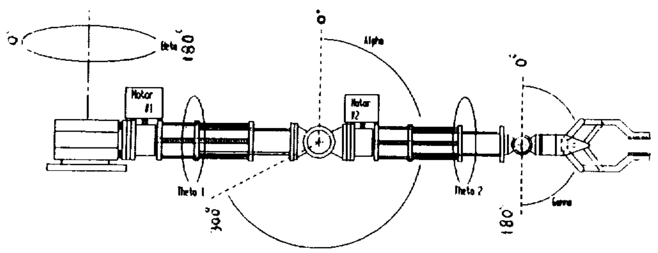


Figure 9: The rotational views of the leg/arm

4.2 Operational Conditions

The only operational procedure of the arm that we will discuss is the one needed for climbing stairs. It is required that an arm support 1/3 of the robot's weight during this procedure. To do so, the arm will be in a straight down position (see Figure 9). The shoulder angle ß must be in the 180° position (provided that the straight up vertical position is defined as the 0° position). The elbow joint must be in the 90° position as well as the hand joint. The hand must be fully closed and both extension units must be extended to the necessary length and locked at that length. Angles Θ_1 and Θ_2 are not important because they rotate axially and do not affect the length. Therefore, the inner locks of both bearing lock systems should be unlocked until the desired extension length is achieved. At that point both the inner and outer locks will be locked.

When the arm is functioning as an arm, and not a supporter, the angles, extensions, and lock positions will vary depending on the situation. We will not cover that situation in this paper.

4.3 Micro-controller Operation and Interface with Arm/Leg Components

In order to perform a micro-controller operation and interface with the arm/leg parts, a major concern is the maximum degree of rotation of the shoulder(ß), the elbow(α), the wrist(gamma), and the twisting of the upper and lower arm extension (Θ_1 and Θ_2). From the arm design shown in Figure 11, it can be seen that the maximum degrees of rotation are as follows:

ß = 360°
α = 300°
Gamma = 180°
$\Theta_1 = 360°$
$\Theta_2 = 360°$

This section contains the basic design of a position/rotation sensing system. The position/rotation information will be processed by the microprocessor to aid in the control of the arm. This system will use a series of simple optical sensors, magnetic sensors, and digital logic.

The arm is designed such that each joint and extension will contain a motor to control its degree of rotation. Note that 12 Volt motors will be used. Since a high supply of amperage is required a H-bridge, consisting of four power transistors, is used to supply power and control the motors.

Two bits are use to control the rotation of the motor. A digital 10 combination represents forward rotation and a 01 allows backward rotation. Using 00 halts the motor. To prevent the power transistors from burning out 11 should never be used.

Position sensing logic that is needed to control the twisting action is defined by the lock configurations of the bearing lock system. These locks will have a digital readings of (1) locked and (0) for unlocked. It was previously stated that for the twisting motion to occur, the inner lock must be locked and the outer lock must be unlocked. The slave processor must register this for the motion to occur. Another input that is needed for position sensing of the twist is the motor's direction. If the motor rotates in a forward direction, the arm will twist clockwise and vice versa if it indicates a backward motion. The degree of twist can be measured by the number of rotations of the motor shaft as detected by a rotational counter that is mounted on the motor shaft. This counter has the ability to detect fractions of turns. The angle can be determined by the gear ratio of the motor gear to the lock shaft of the bearing lock system. For example, if the gear ratio of the motor gear to the lock shaft is n:m, than the angle of rotation (a) of n with respect to m is:

$$\mathbf{a = [360(n/m)]*[\#\ of\ revolutions\ of\ n]} \quad (1)$$

The upper and lower extensions must also contain length controllers. The motor's rotational counter can be used for this. It was stated previously in section three that the extension units will always be functioning whenever the motor is turning. For every rotation of the motor shaft, the extension shaft rotates a fraction of a revolution dependent on the gear ratio. For every rotational motion of the extension shaft, the extension unit extends or contracts some distance depending on the thread pitch. Therefore if the gear ratio, the motor's rotational direction, and thread pitch are known, determination of the extension unit's position is calculated by:

$$\text{Ext. unit dist.} = \#\begin{bmatrix}\text{of turns of}\\ \text{motor gear}\end{bmatrix} \times \begin{bmatrix}\text{gear}\\ \text{ratio}\end{bmatrix} \times \begin{bmatrix}\text{thread}\\ \text{pitch}\end{bmatrix} \quad (2)$$

Equation 2 gives the distance traveled by the extension unit for a certain number of rotations of the motor shaft. However, nothing is said about the original position of the unit. By using the microprocessor logic this problem can be resolved. By calibrating the logic to use the fully contracted shaft as the relative starting point, then all other positions can be calculated by using the equation above.

Position control of the hand is determined by an optical sensor. One optical sensor placed at both extremities of the drive is sufficient. To trigger the sensors the drive nut must have a trigger lip. When the drive nut is against the motor, the hand is closed and a (01) will be sent to the slave processor. When the drive nut is all the way forward, the hand is completely open and a (10) is sent to the slave processor. Any positions between the two extremes will register a (11).

5. FEASIBILITY SIMULATION

Mechanical feasibility was tested in simulation to anticipate possible difficulties in construction. Starting from a downward vertical (standing) home position, the arm is programmed to transfer weight to the wheels, maneuver into positions near an intended object, open and close the hand, and return to home. The simulation plots arm movement in the fourth quadrant of the x and y axes, where point A represents the shoulder, B the first extension unit, C the elbow, D the second extension unit, E the wrist, and F and G the fingertips.

The open_hand and close_hand operations assume a line running between points D and E as the center about which the grasp angle ϕ is measured. The law of cosines is employed to find the orientation of the DE vector with respect to the origin, θ, as calculated in Equation 3 and shown in Figure 10:

$$\theta = acos((c^2-a^2-b^2)/2ab) \ . \qquad (3)$$

Defining the distance from E to F or G as L1, the new F and G coordinates for close_hand are then found as:

$$F_x = G_x = L_1 \cos(\theta) + E_x \ , \qquad (4)$$

and

$$F_y = G_y = L_1 \sin(\theta) + E_y \ , \qquad (5)$$

and for open_hand:

$$\begin{matrix}F_x\\G_x\end{matrix} \mathit{SCALESYM200\}} = L_1 \cos\ (\ \theta \pm \phi\ ) + E_x \qquad (6)$$

and

$$\begin{matrix}F_y\\G_y\end{matrix} \mathit{SCALESYM200\}} = L_1 \sin\ (\ \theta \pm \phi\ ) + E_y \qquad (7)$$

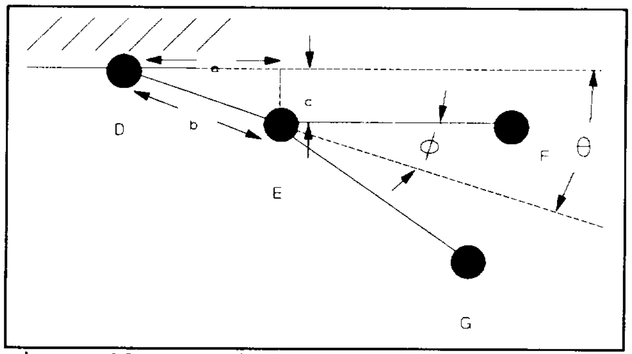


Figure 10: Open_hand and close_hand angles.

Rotations about point A, executed as subroutines swing_left and swing_right, simply rotate all other points about A as origin, as illustrated in Figure 11. Subroutines swing_up and swing_down similarly rotate all points distal to point C as origin. The simulation will redraw the arm when angles rotate or extensions along the AB and DE vectors are user-specified. Safety checks are added to ensure that workspace and robot geometry constraints are not violated, but forces, weights, and frictions are not yet taken into consideration. Coordinates of the sample simulation shown in Figures 12 through 15 were based upon the maximum possible extension of the arm in inches.

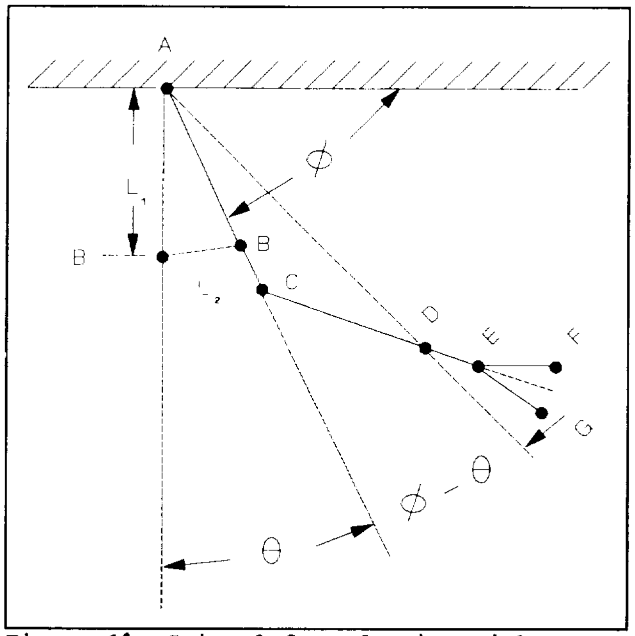


Figure 11: Swing_left and swing_right angles and lengths.

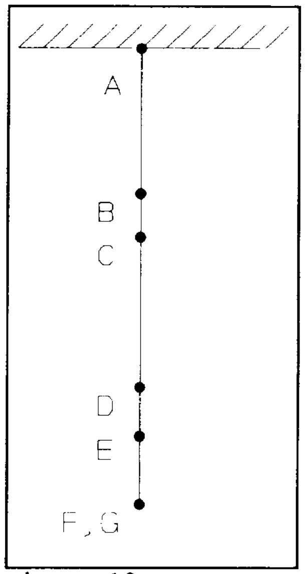


Figure 12: Home Position.

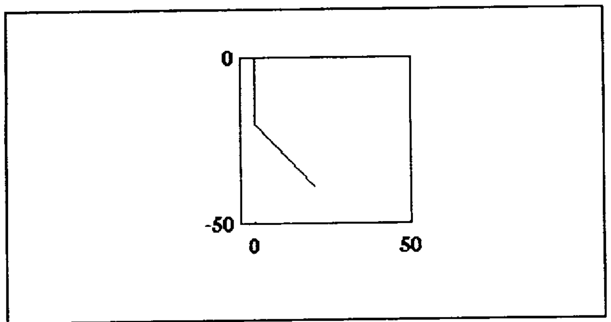


Figure 13: Swing_left of 45 degrees.

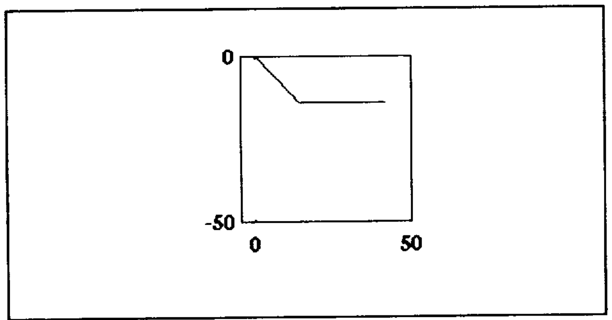


Figure 14: Then a swing_up of 45 degrees.

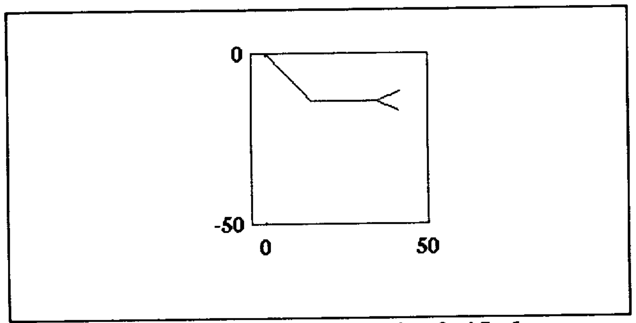


Figure 15: And an open_hand of 45 degrees.

6. CONCLUSIONS

In this paper the structural design and the functional modes of an extended leg/arm used by an autonomous legged/wheeled robot (ODYSSEUS) have been presented. The leg/arm part of the robot plays a very important role by supporting the robot to step over obstacles and climb stairs. The construction of the leg/arm is in progress at the AAAI research lab. The authors wish to thank all the undergraduate students for their work on the ODYSSEUS robot.

REFERENCES

[1] H.Myler,"Design optimization of a six legged walking platform", Int. Conf. on M&S, vol.12,1989, pp.789-792.

[2] W.Chun et.al."Design and construction of a quarter scale model of the walking beam",Int. Conf. on M&S, vol.12,1989, pp.785-788.

[3] Ambler - An autonomous rover for planetary exploration" ,IEEE Computer, June 1989, pp.18-25.

[4] R.A.Brooks,"A robot that walks",in AI in MIT expanded frontiers,ch.25,vol.2,MIT Press,1990,pp.28-39.

[5] M.H.Raibert, M.Chepponis and B.H.Brown, "Running on four legs as through they were one",IEEE Journal on Robotics & Automation,vol.RA-2,No.2,June,70-82,1986

[6] D.J.Todd,"Walking machines",An introduction to legged robots", London: Kogan Page, 1985.

[7] A.C.Klein and S.T.Chung,"Force interaction and allocation for the legs of a walking vehicle",IEEE Journal on Robotics and Automation,vol.RA-3 No.6, Dec. 1987, pp.546-555.

[8] M.H.Raibert, "Legged Robots", in AI in MIT expanded frontiers,ch.30, vol.2, MIT Press, 1990, pp.148-179.

[9] "Microrobotics in Japan", The Japanese robotics project, 1990.

[10]N.Bourbakis,"Design of an autonomous Navigation system",IEEE Control Systems, vol.8,No.5,1988.

[11]N.Bourbakis,"ODYSSEUS-An autonomous walking robot" IEEE Conf. on Intelligent Vehicles June 1992

[12]N.G.Bourbakis,"Real-time path planning of autonomous robots in 2-D unknown dynamic navigation environment",Int. Journal on Intelligent Systems and Robotics,vol.4,pp.333-362, 1991.

[13]N.Bourbakis,"Knowledge-based acquisition during real time path planning in unknown space",in Applications of Learning and Planning methods, WS Pub., March. 1991, pp. 311-331.

[14]M.Maas, N.Bourbakis and A.Mogadazadeh,"Fusion of image and laser sensory data for 3-D modeling", of the free navigation space",Int.NASA Conf. on Intelligent Robotics, March 1994,TX

4027 ,

VAS – A VISION ADVISOR SYSTEM COMBINING AGENTS AND OBJECTED-ORIENTED DATABASES

N94-30532

James L. Eilbert, William Lim, Jay Mendelsohn
Grumman Corporation/CRC
M/S A01-26
Bethpage, NY 11714-3580

p. 10

AIAA-94-1181-CP

Ron Braun and Michael Yearwood
Grumman Corporation/CRC
M/S A04-12
Bethpage, NY 11714-3580

ABSTRACT

A model-based approach to identifying and finding the orientation of non-overlapping parts on a tray has been developed. The part models contain both exact and fuzzy descriptions of part features, and are stored in an object-oriented database. Full identification of the parts involves several interacting tasks each of which is handled by a distinct agent. Using fuzzy information stored in the model allowed part features that were essentially at the noise level to be extracted and used for identification. This was done by focusing attention on the portion of the part where the feature must be found if the current hypothesis of the part ID is correct. In going from one set of parts to another the only thing that needs to be changed is the database of part models. This work is part of an effort in developing a Vision Advisor System (VAS) that combines agents and objected-oriented databases.

INTRODUCTION

The bulkheads of Grumman aircraft, including the E2C, are assembled using a visually guided robot cell, called the Flexible Assembly System (FAS). Parts are laid out on a tray with the surface of the part that is to be attached to the bulkhead against the tray. Each part has a flange that is perpendicular to the tray. The robot receives information about the position and orientation of parts on the tray from a 2-D vision system which looks directly down on the tray, located about 6 feet away from the cameras. This means that the flanges that the vision system must locate are viewed edge on. FAS uses the coordinates supplied by the 2-D system to move a robot arm to the designated pickup point on a part which is always located on the flange. Once the arm is in position, the part is picked up at the pickup point. A 3-D camera with very limited range is used to find the positional error between a marker hole on the part and a reference hole on the gripper. Correcting this error allows the robot to determine the position of the part on the gripper accurately for placement on the bulkhead. After the part is placed against the bulkhead, the robot rivets it in place.

The Vision Advisory System (VAS) reported in this paper concerns the identification of parts and the location of their flanges using tray images such as the one shown in Fig. 1. Our goal is to make VAS an autonomous visual recognition system where the only change needed when the robot begins work on a new part set is a database describing the new parts. VAS is currently in the evaluation stage. It runs on a Macintosh 2fx connected via NFS to a Sun computer which runs the old 2-D vision system. Thus, it operates on the existing FAS manufacturing system, in parallel to the older, less-than-satisfactory 2-D vision system. The current

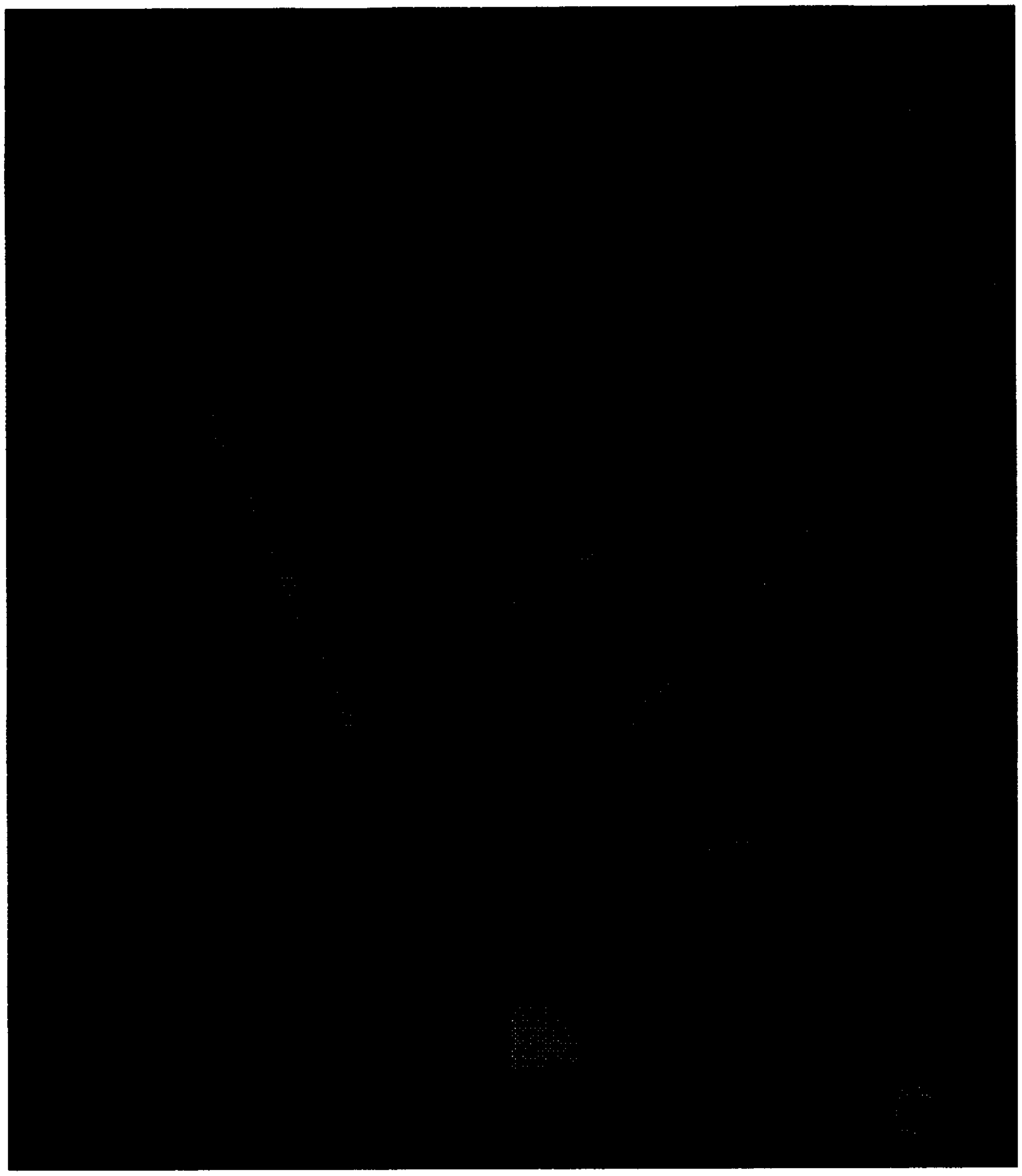


Fig. 1. A Typical tray image in which the fiducial marks in the corners were found and used to calibrate distance.

role of VAS is to provide "advice" regarding part identification and flange location to the existing system so a better decision can be made with regard to where the part pickup point is.

SYSTEM CONSTRAINTS

Although the parts never overlap or even touch one another the problem being addressed is made difficult by the similarity of some parts. Often the only difference between parts is the position of *features* on the sides of the parts, such as *bumps* and *notches.* These features can be viewed as convex or concave imperfections in the normally smooth and straight sides of the parts. Since the relative positions of the flange and the part's features are stored in a database, finding a feature solve the flange location task as well as the part ID task. The previous 2-D vision system, built in the mid-1980s, used a coarse shape description that generally ignored these small features. In fact, it is the inability of that system to deal with small features that lead to the the current upgrade.

Since the robot was already in production when our task began, we were constrained to use the components of the existing system. As a result, the features (i.e. bumps or notches) that can distinguish parts or indicate the flange location are only 1-2 pixels in height or depth. Due to their small size, the precise position and extent of features on the parts cannot be determined.

In addition to the small features there are two previously unused sources of information that could simplify the solution to the 2-D recognition problem, namely shadows and context information. In cases where there is a clear shadow, it is possible to find the flange on a part even if it has no features at all. However, due to assymmetries in the lighting, shadows that are reliable predictors of the flange in one orientation cannot be seen at all in others.

Context information, i.e. the part descriptions in the database and the history of the current assembly session, can be used to augment and simplify the recognition process. For example, the knowledge of what parts have already been removed from the tray, can allow a part to be trivially identified if all of the parts similar to it have already been removed. The search for a feature in a small region under the assumption that some candidate model corresponds to the true part is also an example of the use of context information.

Since rule-based reasoning is relatively expensive in terms of the time it takes, the system avoids reasoning about context when possible. In cases where a sequence of inexpensive image operators leads to unambiguous results, the system does not do any additional reasoning. However, when there is ambiguity the system is able to reason about a part's ID or flange location using context information or even to decide that additional information must be extracted from the tray image.

The goal of only switching the physical descriptions of the parts making up the data set is not possible unless the system has all the image operators it will ever need. In particular, it assumes that image operators exist which allow any two features that can be found on any part to be distinguished. This is a not possible when you do not know what the parts in future sets will look like. To deal with this type of novelty the system must be able to "learn" to descriminate the new features from all existing features. In order to meet these requirements the system we propose must be able to do a limited amount of planning, learning, and high level representation.

A PROPOSED VISION ARCHITECTURE COMBINING AGENTS AND OBJECT-ORIENTED DATABASES

Following Minsky's (15) Society of Mind paradigm, researchers in a number of fields have begun proposing agent architectures. The emerging interest in Distributed AI (14,11,10,18) and in distributed control systems (5) has literally forced researchers to look at agent architectures of various types. However, researchers looking at autonomous systems that have multiple goals or drives and operate in several domains have been equally drawn to agents (1,2,3) . The solution to the FAS 2-D vision problem discussed above requires an autonomous system carrying several tasks with several domains in which it must be knowledgable (i.e. tray images, databases, robot arm coordinates). This suggested that the 2-D vision system could be naturally implemented as an agent architecture with a set of autonomous agents interacting with each other and an object-oriented database. In fact, the agent architecture chosen is a simplified version of an architecture originally developed for Automatic Target Recognition tasks (6). In this paper, we describe the building blocks being implemented to support such an architecture. For example the agents and the object-oriented database are implemented in CLOS (LISP), while the basic image operators are written in C. Note that at present the agents we have implemented do not have the full capability of the agents we envision, since the full support structure for the agent architecture is still being implemented.

Agents

The particular agent architecture used was developed at Grumman (13), and is an integration of the object-oriented programming paradigm and expert systems. Agents are both local experts and objects. Two basic classes of agents are available in the architecture: agents that manage a behavior and agents that plan to satisfy a goal or drive using a sequence of behaviors. Since the concepts of behaviors and plans play a central role in our approach and the terms are used in such a wide variety of ways, we provide the following list of definitions.

• *Behaviors* are a triple of actions, i.e. {activation, execution, termination} .
• *Routines* are control algorithms or a sequence of mappings between sensations that correspond to the execution portion of a behavior.
• A *domain* is a set of similar environments (ex. trays 1-5 of the FAS robot cell). In general, domain is a set of environments that is "sufficiently similar" to a prototype or a set of examplars, where both the prototype and the measure of similarity must be included in the definition. A routine may have a domain of validity associated with it.
• A *landmark state* is a description of the relation between the robot and important objects in the world.
• A *plan* is triple of events {recognition of start state, plan execution, recognition of goal state}.
• An *intention* is a sequence of mappings between landmark states that correspond to the execution portion of a plan.
• A *plan domain* is a set of similar situations (ex. part in reach on any of trays 1-5 of the FAS robot cell). Again a meaningful definition requires a prototype or examplars, and a measure of similarity must be included in the definition.

The behavior-managing agents are concerned with the moment to moment interaction between an entity and its environment, and the interpretation of sensation. The behavior-managing agent stores a set of routines plus information about its domain of applicability. In addition, it must be able to receive and store information about starting and stopping states from the planning agents with which it communicates. It must be able to translate these states into predictions about the corresponding sensations which it will actually detect. Behaviors have been developed for finding part boundaries, long-lines on the boundaries, and the bounding rectangle; and for detecting bumps, dents, and shadows. A behavior managing agent for "bump detection in the middle of a part" would decide when and where the search should take place, as well as when the search has succeeded and when it has failed.

The planning agents are concerned with "landmark states" and how to move between them. Each step in a plan must correspond to the resulting state change that occurs when a behavior is executed (12). Planners are incapable of operating in "real-time" since they do not have access to the real world through sensations. However, they may know what sequence of landmark states they will pass through before they need to stop planning. The planner stores a set of intentions plus information about its planning domain. In addition, it must be able to receive and store information about the current states from the planning agents with which it communicates. It must be able to compare these states with expected states to determine if the plan is working. A planning agent whose intention is to "locate flanges", would decide what combination of shadows and features to use in finding the flange and how to weight them. It would also send activation and termination states to the appropriate managing agent.

Agents combat the traditional brittleness of expert systems, associated with operating in too large a domain, by having many task specific behavior-managing agents that are competent in small domains and much fewer planning agents that monitor their applicability and performance. Like other objects, agents can communicate by passing messages and they also have dedicated communication lines to other agents. There are also communication lines to the objects in the object-oriented database.

The Object-Oriented Database

Model-based vision requires data bases to store both models and various types of information obtained from the actual images. Most of researchers who have looked at the image database problem have advocated using object-oriented techniques even when their specifications are significantly different (8). The object-oriented database utilized for VAS must store two type of information, in addition to the description of parts (in terms of sensation): spatial relationship of parts on the tray over time, and the plans that have successfully been used to do the major tasks. We have described elsewhere a high level representation consisting of the grid map, a graph of landmark states and a set of part descriptions that can organize this type of information (6). The grid map describes a particular environment and the spatial relationship of objects within it. In this case, the environment is a tray and the the relationships are among the parts, fiducial marks, and clutter. The grid map is a bird's eye view constructed from a set of scenes that shows the relative positions of the important objects, but little detail of their internal structure. The graph of landmark states describes the plans that are valid in a given environment. The landmark graph is a network of (state) objects as is a standard AI semantic net (16). However, the nodes of the landmark graph are connected to each other by plans for moving between states, rather than "ISA, PARTOF, or PROPERTYOF" links. Note that not all state nodes in the landmark graph involve "physical landmarks", some nodes involve temporary objects and are labelled as such. These two maps capture the spatial relationships and the plans learned for moving around an environment, and have most of the properties attributed to cognitive maps in living animals (17,7).

Discrimnation Net

Discrimination nets are a simple AI technique for classifying objects based on a set of common properties with two or a small number of values (4). The use of fuzzy properties to describe parts makes it possible to use a discrimination net to classify the part models in a database. When parts with very similar sizes and shapes must be identified, it is important to keep all reasonable candidates until a final discrimination is made. The discrimination net does exactly this. To use the discrimination net one would make a list of the properties of an image-object and run them through the net. Each property is used to pick a direction in the net until a leaf node is reached and a part ID is returned. If a leaf node is not reached a small number of candidates are returned. The discrimination net actually consists of a sequence of keys and linked lists. If the list (SHORT MEDIUM ((BUMP .1) (NOTCH . 0))) were submitted to the net, the relevant part of which is illustrated in Fig. 2, then both part 787 and 7101 would be returned. The decision of which of these parts is being examined would require looking at rough feature position or the quantitative measures of length or width. The issue of setting model-based matching criteria is a difficult problem in general (9), but our images have simple backgrounds and good part background separation which simplifies things.

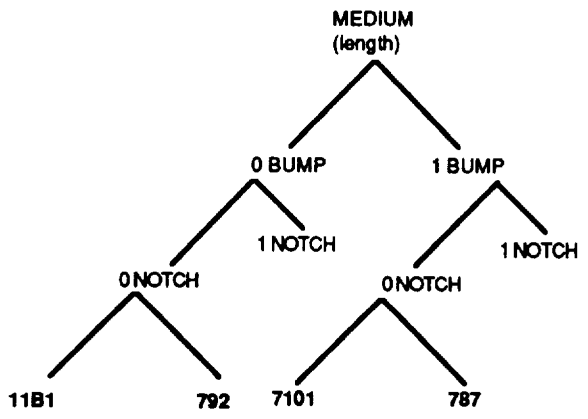


Fig. 2. A sample portion of a discrimination net.

THE CURRENT SYSTEM

There are six major tasks for the 2-D vision system, i.e. database completion, part outlining, long-line finding, part ID, flange location, and the sending of pickup coordinates to the robot arm. These goals each have a planning agent associated with them. Together these agents through their interactions carry the part ID and flange location tasks that is the real purpose of the 2-D vision system. Overall processing is initiated by the operator with a Lisp function called "run-FAS" which takes a database of part-models as an argument. The system then runs till the following midnight when it reports its results.

When the system first encounters a new data set the database completion agent is activated, it moves through each part model and extracts fuzzy descriptions of the length, width, and features. This information is stored in appropriate slots of the part models. The following is a typical part model description from the database where the slots in **bold** are automatically filled in by database completion agent:

```
;;; Part 715
(setq 715
 (make-instance 'part-model
          :model-name '715
          :home-tray nil
          :home-bank nil
          :surface-list nil
          :center-of-mass nil
          :part-length 23.20
          :fuzzy-length nil
          :part-width 0.631
          :len-wid-ratio nil
          :fuzzy-width nil
          :bump-list '((0.0 .625 .128)
                       (6.75 7.39 .14)
                       (13.38 14.06 .14)
                       (20.63 21.49 .14))
          :hook-list nil
          :needle-list nil
          :notch-list nil
          :tail-list nil
          :easy-features nil
          :grip nil
          :flange-height 0.638
          :major-flange-bumps nil
          :flange-shape 'L
          :similar-part-list nil
          :action-list nil
          :action-code nil
          :group *load-group*))
```

This agent then builds a discrimination net for all parts in the data set and stores them in the experiment data object. When the database is complete, it sends a message which activates the part outlining agent.

The part outlining agent then monitors a working image directory to see if a tray image has been captured. It takes a pair of images to cover the entire tray. The basic algorithm that the part outlining agent uses consists of adaptive thresholding, morphological smoothing, and a boundary following procedure. The results of this process are shown in Fig. 3. When the part outlining agent completes its task it activates the long-line finding agent.

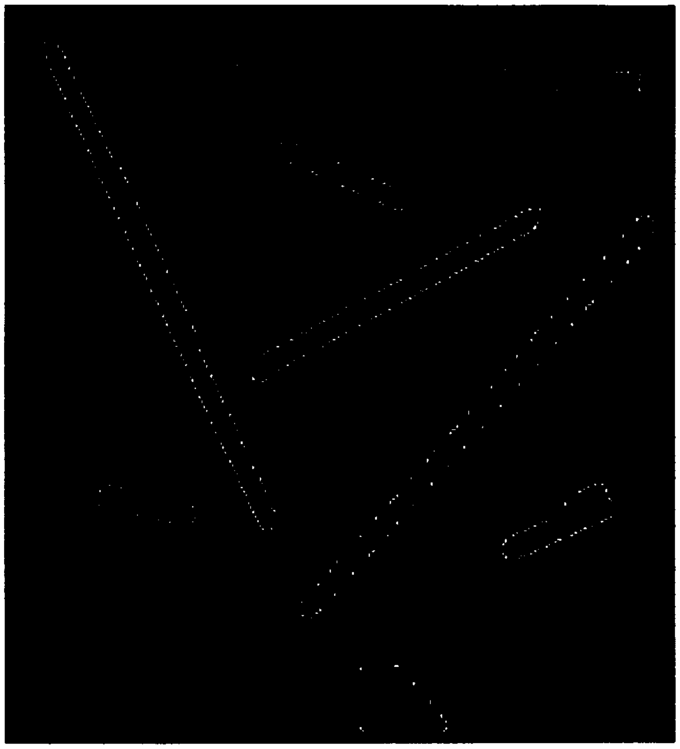


Fig. 3. Results of the Part Outlining Task

All of the parts with which we have worked are long and thin. If parts that do not have this general shape appear in the database a new agent that finds their bounding rectangle will have to be developed. The long-line finding agent uses a simplified Hough transform to find candidate line segments in the outline that may belong to the long lines. It then uses a mean square error best fit line on the line segments that are sufficiently similar. Fig. 4 shows the long-lines found for each part, plus the bounding rectangle for the long part on the left. If the long-line agent completes its task, it activates the part ID agent. Note that all of the processing up to this point runs automatically with only basic checks for failure. All of the information is stored in a data object called a tray. A partial listing of a filled tray follows:

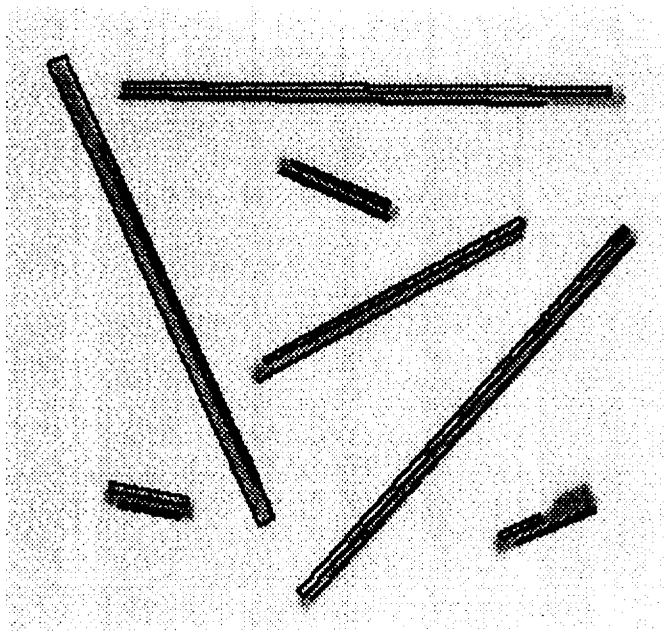


Figure 4: The Long-lines Found for the Parts (Color reversed for clearer graphic display.)

```
#D(TRAY
  TRAY-NAME
"Images:Shading:N_shortN.8bits"
  IMAGE-OBJS
  (#562=#D(IMAGE-OBJ
        PART-NAME NIL
        TRAY-NAME
           "Images:Shading:N_shortN.8bits"
        TRAY-COORD-CENTER (199 . 294)
        PART-AXIS NIL
        PART-LENGTH 252.906525
        PART-WIDTH 19.008202
        LEN-WID-RATIO NIL
        TRAY-COORD-ANGLE 0.0
        BUMP-COUNT 0
        BUMP-LIST ((NIL NIL NIL)
                   (NIL NIL NIL))
        NOTCH-COUNT 4
        NOTCH-LIST ((NIL (((312 . 287)
                           (313 . 288)
                    1.756594313390609))
                  (((89 . 286) (88 . 288)
                    3.1204771710092984)))
               ((((138 . 303) (129 . 303)
                 1.30427164452175))
                NIL
               (((125 . 303) (88 . 300)
                 4.814981468417682))))
```

```
ZOOMABLE NIL
INSIDE-INTENSITY-AVE
                ((73.0 11.346012 1139)
                 (85.0 11.083988 910))
INSIDE-GRAD-AVE NIL
OUTSIDE-INTENSITY-AVE
                ((20.0 4.298134 769)
                 (26.0 11.714762 994))
LONGEST-LINES (#D(ANGLINE
               ANGLE 0.00162448
               LN-LENGTH 222.001331
               Y-INTER NIL
               END1 (312 . 285)
               END2 (90 . 285)
               GROUP NIL
               )
             #D(ANGLINE
               ANGLE -0.0124623417
               LN-LENGTH 187.010402
               Y-INTER NIL
               END1 (313 . 302)
               END2 (126 . 304)
               GROUP NIL
               ))
     BOUNDING-RECT ((313 . 285)
               (313 . 302)
               (88 . 305)
               (88 . 285)) ...
```

The basic approach taken by the part ID agent is to use the fuzzy discrimination net describing the gross characteristics of all the parts in the current database. The part ID agent takes the information about a part which was found by outlining and long-line finding agents and fills in the fuzzy slots for its image part in the object-oriented database. The fuzzy information from the part in the current tray image is turned into a list and run through the discrimination net. A short list of candidate parts is returned.

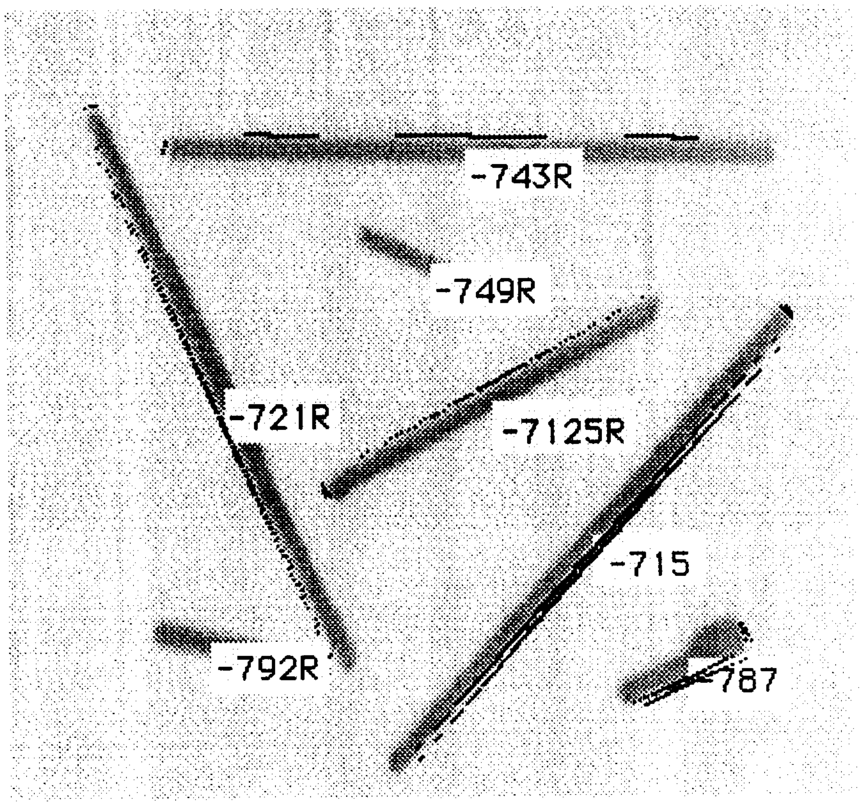


Figure 5: The Part IDs and the Flanges Found (Color reversed for clearer graphic display.)

For example, consider getting the ID of a part centered at (252 . 320) on the tray. It has no bumps nor notches and its length 5.15 (SHORT) and width 0.75 (FAT). The candidates parts with the correct feature list are:

```
((SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART119)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART70)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART69)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART55)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART49)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART59)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART82)
 (SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART81))
```

The candidates for the object are reduced to PART55, PART70, and PART59. If a definite ID cannot be made based on differences in the length and width, then small features are sought in particular places. Finally, the presence of IDed image parts activates the flange finding agent.

Initially, the flange finding agent ignores the image part IDs and executes a shadow finding routine. By setting the criterion for finding a shadow high we can force all classifications to be correct or unkown. If shadowing does not yield an answer then small features are sought where they should be based on the part ID. Fig. 5 shows the final IDs and flanges found. All IDs are correct and five of seven flanges were found correctly. In the two cases where the flange was not located correctly, i.e. 749 and 792, VAS reported that it could not find the flange rather than making an error. Note that neither part had features, and that human observers were also unable to locate those flanges. Of the five flanges found, 743 was located based on its shadow, while the other four were located based on finding features.

Each of these agents have contingency plans that are implemented when basic algorithms fail in particular ways. For example, if the part IDer does not come up with a unique ID, it will send a request to the database to give it the lengths, widths and the approximate location of the features on each of its candidate models. Agents also communicate indirectly with each other through the tray and image-objects. Since everything of use to any of the other agents is recorded on these objects, agents do not need to know which agent calculated a piece of information in order to use it. Thus, an agent will always check the database to see if a piece of information that it needs is available before it tries to extract it directly, or sends a request to another agent. Controlling the communication among agents is one of the major challenges of agent architectures, and is still being studied for VAS.

CONCLUSIONS

A model-based approach which uses fuzzy descriptions of part features for classification and an object-oriented database of parts has been developed. A variety of image processing techniques have been combined to find the information needed to do identifcation and flange location, i.e. length, width, small bumps and dents, and shadows on the parts. It was possible to decompose the overall task into a set of modular tasks that interact and fail in specific ways. An agent architecture has been developed that takes advantage of the modularity in this multi-task and multi-environment domain. The success of a vision architecture initially developed for a wholely different application, i.e. automatic target recognition give us hope that the agent approach to autonomous vision problems is a general one.

One final point is that with better cameras and lighting many of the problems that proved very stuborn in VAS would never have come up. However, good design is hard to do when the scope of the problems the system will face are not known in advance.

REFERENCES

1- Agre, P., Chapman, D. (1987) Pengi: An implementation of a theory of activity. Proc 6th National Conf on Artificial Intelligence, AAAI-6.
2- Anderson, T.L., Donath, M. (1988) A computational structure for enforcing reactive behavior in a mobile robot.

SPIE V. 1007: Mobile Robots III. Boston, MA.

3- Brooks, R.A. (1986) A robust layered control systemfor a mobile robot. IEEE J. Robotics and Automation, RA-2. 14-23.

4- Charniak,E., Riesbeck, C.K. McDermott, D.V., & Meehan, J.R. (1987) Artificial Intelligence Programming. 2nd ed. Hillsdale, NJ: Lawrence Erlbaum Assoc.

5- Cui, X., Shin, K.G. (1991) Intelligent coordination of multiple systems with neural networks. IEEE Man, Sys, Cybernetic, 21(6):1488-97.

6- Eilbert, J.L., Mendelsohn, J. (1992) Organizing and using context information for ATR. pp. 303-312. Proc. Second Automatic Target Recognizer System & Technology Conf.

7- Gallistel, C.R. (1990) The Organization of Learning. Cambridge, MA: Bradford Book.

8- Goodman, A.M., Haralick, R.M., Shapiro, L.G. (1990) Knowledge-based computer vision- integrated programming language and data management system. IEEE Computer 22(12): 43-58.

9- Grimson W.E.L., Huttenlocher, D.P. (1991) On the verification of hypothesized matches in model-based recognition.

10- Hewitt, C., Inman, J. (1991) DAI betwixt and between: From intelligent agents to to open systems science. IEEE Man, Sys, Cybernetic, 21(6):1409-19.

11- Kaebling, L.P., Rosenschien, S. R. (1990) Action and planning in embedded agents. pp. 35-48. In: Maes, P., Designing Autonomous Agents. Cambridge, MA: Bradford Book.

12- Lim, W., Eilbert, J.L. (1990) Plan-Behavior Interaction in Autonomous Navigation. SPIE V. 1388: Mobile Robots V:464-475. Boston, MA.

13- Lim, W., Verzulli, J. (1990) SAL -- A language for developing an agent-based architecture for mobile robots. pp. 285-296. SPIE VII. 1831: Mobile Robots VII. Boston, MA.

14- Maes, P. (1990) Designing Autonomous Agents. Cambridge, MA: Bradford Book.

15- Minsky, M. (1986) The Society of Mind. NY: Simon & Schuster.

16- Nilsson ,N.J. (1980) Principles of Artificial Intelligence. Los Atlos, CA:Morgan Kauffmann.

17- Tolman, E.C. (1948) Cognitive maps in rats and men. Psychol. Rev. 55:189-208.

18- Wong, S.T.C. (1993) COSMO: A communication scheme for cooperative knowledge-based systems. IEEE Man, Sys, Cybernetic, 23(3):809-824.

6028

AIAA-94-1182-CP

APPLICATION OF THE MODULAR AUTOMATED RECONFIGURABLE ASSEMBLY SYSTEM (MARAS) CONCEPT TO ADAPTABLE VISION GAUGING AND PARTS FEEDING

N94-30533

Andre Bernard By[a] and Ken Caron[b]
Department of Mechanical Engineering, Tufts University
Medford, Massachusetts

Michael Rothenberg[c] and Vic Sales[d]
Productivity Technologies Incorporated
Sunnyvale, California

-/-6-)

)

P. 9

Abstract

This paper presents the first phase results of a collaborative effort between university researchers and a flexible assembly systems integrator to implement a comprehensive modular approach to flexible assembly automation. This approach, named MARAS (Modular Automated Reconfigurable Assembly System), has been structured to support multiple levels of modularity in terms of both physical components and system control functions.

The initial focus of the MARAS development has been on parts gauging and feeding operations for cylinder lock assembly. This phase is nearing completion and has resulted in the development of a highly configurable system for vision gauging functions on a wide range of small components (2 mm to 100 mm in size). The reconfigurable concepts implemented in this adaptive Vision Gauging Module (VGM) are now being extended to applicable aspects of the singulating, selecting, and orienting functions required for the flexible feeding of similar mechanical components and assemblies.

1.0 Contemporary Flexible Assembly Technology

Andreasen, Kahler, and Lund[1] have defined assembly processes as composed of three main stages: handling, composing and checking. These three stages can in turn be subdivided into storage, transport, and positioning functions. Another view of the assembly process is to define it in terms of operations related to workpart gauging, feeding, gripping, and fixturing.

Independent of the classification approach used for assembly processes, workparts must generally be properly gauged or tested, fed, oriented, and held for the assembly function to be a success. Many researchers have attempted to provide suitable analytical approaches to model this processing of workparts, often by looking at one function (such as trajectory or motion planning, collision avoidance, parts insertion, etc.) in high detail. Others have noted the commonality between many of these functions and attempted to leverage this to define the separate problems in a common context.

[a] Research Associate/Lecturer, Mech Engineering, Tufts
[b] Graduate Student, Mechanical Engineering, Tufts
[c] Project Engineer, Productivity Technologies
[d] Project Manager, Productivity Technologies

For example, Natarajan[2] observed the duality between the motion planning problem and the problem of designing a feeder for orienting a workpart from some arbitrary initial orientation I to a final orientation G. In principle, an algorithm that would facilitate automatic design of parts feeders based on CAD/CAM representations of workparts should also be able to benefit from previous developments in workpart grasp modelling. Similar techniques should also be applicable to the problems of flexible fixturing system synthesis and gauging function definition and design.

Unfortunately, the practical industrial technologies and tools for making these support operations adaptable from process to process (or part type to part type) are currently very limited. Typical so-called flexible assembly systems (FAS) in use today are often fairly flexible in terms of potential workpart trajectories, yet relatively primitive in terms of easily or automatically adapting to the various aligning, gripping, and fixturing needs of different workparts or processes. These flexible assembly systems are often little more than a robot surrounded by a set of fixed tooling that is programmed once and left to run for several months or years until new production needs dictate system retooling. The potential advantages of flexible automation are thus hardly realized in this scenario.

Machine vision subsystems, quick change tooling modules, and various advances in off-line programming/simulation systems[3, 4] have been suggested as the essential breakthroughs that will pave the way for a proliferation of cost effective and truly flexible (agile) assembly systems. However, most automated assembly systems being implemented today still employ primarily fixed tooling for the actual grippers, fixturing, vision gauging system components (optics, lighting, mountings, etc.), and parts feeding/orienting/guiding functionality. Machine vision systems have become easier to setup and program yet the required support equipment for parts presentation and illumination still entails significant custom design and fabrication. Quick change tooling modules are typically used to simply swap one fixed piece of end of arm tooling for another.

Off-line programming/planning/simulation systems can improve the efficiency of designing and programming automated assembly systems. Alternative system approaches and assembly task strategies can be quickly evaluated and compared prior to the fabrication of a proposed system. However, the resulting assembly system designs are not necessarily more flexible. Further, assembly system programming changes or adaptations are still done off-line and not local to the actual assembly system controller. This minimizes the ability for the assembly system to automatically reconfigure itself under local control. Thus, the resulting assembly system is not able to as easily and quickly adapt itself to required changes in production schedules or capacity balance. The ability for such dynamic reconfiguration is likely to be an increasingly important feature as assembly systems become more flexible.

2.0 Application of Analytical Tools to Industry Practice

Although significant, most of the leading edge analytical advances in flexible assembly over the last several years have yet to be applied to solve practical real-world manufacturing requirements. This is not entirely surprising since leading edge analytical developments, by definition, are not directly amenable to industry application. Another potentially significant factor is the relatively limited practical collaboration between researchers in the academic community and system integrators and end users in industry.

Academic research tends to focus on issues that are more abstract and provide long term benefit to the state of the art. Innovators in industry tend to focus on more near term and precise objectives such as delivering a working assembly system next quarter that will operate with 99.5% up-time and support N variations on a set family of workparts with setup changeover not to exceed M hours. This difference in objectives and motivations can tend to preclude meaningful collaboration.

There are, however, significant potential advantages to such collaborations. The difference in approach by academics and industry can provide new perspectives to each group. It is generally recognized that collaborative teams composed of individuals with diverse backgrounds can act to improve both the effectiveness and rate of innovation[5,6]. There are two essential ingredients to achieving this potential in practice:

1. An appropriate framework of project objectives that emphasizes goals and results which are clear to all contributors.

2. A suitable means of monitoring and managing the progress of the team towards these objectives.

It is not the intent of this paper to investigate or pursue the validity of these observations at length. This subject has been and continues to be the focus of substantial research and study by others. However, we will use these as a guide in defining a framework for the development of an integrated approach to support the essential assembly process functions in a truly flexible assembly system.

3.0 The MARAS Concept

From the above, the new approach to flexible automated assembly development should foster effective collaboration and synergy between contributors in both academia and industry. It is also important that it facilitate the adaptation and extension of appropriate analytical tools to real world applications. Towards these ends, Table 1 defines primary characteristics of the new approach, named MARAS (Modular Automated Reconfigurable Assembly System).

Table 1

Primary MARAS Concept Features

1.	Use building block approach for mechanical modules and subelements.
2.	Employ unified analytical models, scalable from basic to advanced capability.
3.	Use object-oriented representations of physical elements (including actuators, passive components, and sensors) as well as software control functions.
4.	Use building block elements for modelling and control functions.
5.	Initial emphasis on a specific range of parts that is small enough to be practical yet with enough general features so as not to be trivial.

MARAS has some similarity to other contemporary reconfigurable system concepts in that it emphasizes a modular or building block approach to implementing flexible assembly systems. However, MARAS is structured to emphasize multi-level modularity for both system physical components and related system control software.

At the physical level, MARAS extends some of the modular concepts for flexible fixturing defined by Asada and By[7,8] and borrows from other modular approaches such as the RobotWorld modular robotic station base concept of Scheinman[9] and the Carnegie Mellon Reconfigurable Modular Manipulator[10]. The MARAS concept extends the approach of flexible fixture system synthesis based on combining appropriate fixturing subelements (fixels) to combine similar families of physical elements to support the other fundamental functions required in assembly:

1. Gaugels (physical elements that are combined together to form Vision Gauging Modules or VGMs).

2. Feedels (physical elements that are combined together to form Adaptable Feeding Modules or AFMs).

3. Grippels (physical elements that are combined together to form Generalized Gripper Modules or GGMs).

MARAS specifies that these building block elements will be represented in an object-oriented fashion to facilitate the development of a unified set of analytical tools. Further, the use of object-oriented representations for these mechanical elements can potentially act as an integrating common reference frame for machine element designers, analysis model developers, and software systems engineers. Machine designers can create a database of mechanical element definitions that can be matched and integrated as needed to meet the requirements of different workparts and assembly support functions.

Analytical modelers will initially apply group technology to classify these building block elements. This includes extended definition parameters to categorize and describe the elements in terms of how they can be controlled or used in conjunction with other elements to support specific functions and/or specific types of workparts. These categorizations will form the basic of applied modelling tools to be developed for prediction of the performance of the initial set of elements and associated design derivatives or improvements.

Software system engineers will utilize the element representations to support the development of efficient software modules or objects for the monitoring and control of the MARAS assembly system modules. This will aid in the translation of analytical modelling tools from theory to practice. Both will be based on the same data object representations. Control and sequencing functions to be performed will also be defined as generalized objects and methods to further assist in the development of a unified modelling and control system.

The use of a common building block definition system for machine designers, analytical modelers, and software developers will improve understanding between the various contributors. It should also lead to more focused innovation. The common representation will allow advances or improvements in one area, such as modelling tools development, to be more readily applicable to other aspects of the concept as it evolves. Developments in each area can start at a basic level and be gradually scaled with time to be more sophisticated in scope and robustness.

4.0 Phase I MARAS Focus

A specific set of small parts, cylinder lock components, was selected for the first phase of MARAS system development. These components range in size from approximately 1 mm to 25 mm in length. Some of the parts are mostly planar while others are cylindrical or more complex in shape. This provides a reasonable variety of shapes, aspect ratios, and details such that the part family includes a number of aspects found in other small parts.

Two fundamental functions of the assembly process were selected for this initial phase: vision gauging and parts feeding/orienting. The definition of an initial basic set of modular elements and corresponding modular control approaches for these functions was the primary objective. The development of corresponding analytical tools for these elements is currently in progress but is not part of the scope of this paper.

This initial focus has resulted in the development of a working version of one subsystem of a practical MARAS system: the Vision Gauging Module (VGM). Preliminary physical building block element definitions for another important subsystem, the Adaptable Feeding Module (AFM), have also been completed.

The VGM is a reconfigurable subsystem for vision inspection of small (1 mm to 100 mm) mechanical components. Fixturing, illumination, optics, and other required physical elements of the VGM (gaugels) have been designed to address different part family applications with little or no mechanical setup change. A corresponding set of software modules has been defined and developed to support automated execution of system changeover to support new part family inspection operations with object-oriented definition of system operations. Here, the VGM can be reconfigured on-line to perform entirely new gauging functions based on a device configuration database downloaded to the VGM controller by a supervisory controller with links to parametric CAD representations of the parts to be gauged.

The AFM is a similar subsystem and approach for adaptable parts feeding and orienting. Geometric analysis of parts to be supported by the system will define guiding checking/inspection elements (called feedels) from a generalized family. The active control of the AFM will also be driven by the geometric representations. As with the VGM, the AFM will be reconfigured on-line to perform entirely new feeding functions based on a device configuration database downloaded to the AFM controller.

5.0 Phase I Parts Description

Figure 1 provides a simple side view of the primary components to be assembled to produce a typical cylinder lock. The plug is a rotating cylindrical component that is turned by the key within the cylindrical base of the body. The driver, attached to the plug by the cap, is the component that activates the lock latching mechanism. The plug will only rotate if the key's notches match the heights of the corresponding base pins installed in the cylinder lock assembly.

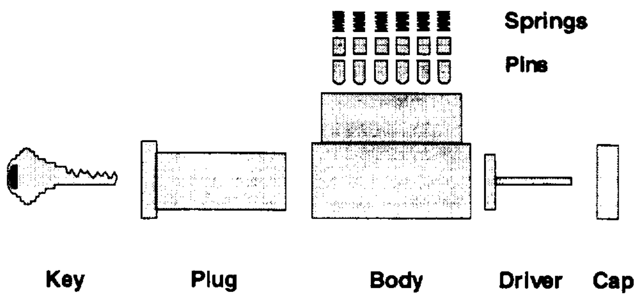


Figure 1 - Cylinder Assembly Components

The key, plug, and body are each approximately 25 mm (1 inch) in length. Other part dimensions are roughly to scale as shown in Figure 1. The part variations and gauging requirements are extensive, as indicated in Table 2. Each of 30 key types employs a unique key blade cross section with different sets of dimensions and tolerances for each type. Further, each key type is available in either 5 or 6 pin variations. The key blade cross section variations also apply to the plugs, since the keys are inserted into the plugs to operate the cylinder.

Table 2

Part Variations

Part	**Variations**	**Gauging Requirements**
Keys	30 types, 2 lengths	8 dimensions
Plugs	30 types, 2 lengths	25 dimensions
Bodies	2 types, 2 lengths	25 dimensions
Pins	2 types, 13 lengths	5 dimensions
Springs	2 types	3 dimensions
Drivers	8 types	3 dimensions
Caps	2 types	3 dimensions

5.1 Parts Handling, Orienting, and Gauging Requirements

Both the plugs and bodies include a high number of dimensions to be checked. This includes pin hole locations and alignment plus face and tail details such as key slot alignment, concentricity, and body "tang" alignment. The tang of the body is the vertical block that holds the top pins and springs. Two lengths of bodies and plugs are required, to support both 5 and 6 pin variations.

Two types of pins (bottom pins and top pins) must be supported. Bottom pins alone have 10 unique lengths. Radius of nose curvature for each end of the base pins is different, since the leading (bottom) edge of the pin must be very narrow to mate effectively with the key notches and the top of the top of the pin must be relatively flat.

Table 3

Part Orienting Requirements

Part	Source	Orienting Requirements
Keys	Bulk	Vertical
Plugs	Bulk	Vertical
Bodies	Pre-oriented	Vertical
Pins	Bulk	Lead edge down
Springs	Bulk	Vertical
Drivers	Bulk	Vertical, tail up
Caps	Bulk	Threads down

Variations for caps and springs are fairly minor (one or two different dimensions) but drivers come in a wide range of styles to support different types of latching mechanisms.

Table 3 summarizes the source of the parts to be supported plus the final orientation requirements for use by the robotic flexible assembly stations. Excluding cylinder bodies, all parts are originally without any orientation and are located in bulk bins. Many parts are also fabricated both in-plant and externally.

The source of the incoming parts is one of many additional issues to be considered in defining and implementing a gauging and feeding system approach. These issues also include:

1. Parts may need to be assembled both in plant and externally.
2. Gauging is required between fabrication operations or steps for some parts.
3. The production rate approximates 1 part/second.
4. Low capital is available for project implementation.
5. The required implementation schedule is short.

6.0 The Modular Inspection/Palletization Cell

In applying MARAS to the cylinder lock parts gauging and orienting application, the following system design constraints were defined:

1. Gauged and oriented parts will be loaded to pallets (to address both in-plant assembly and external assembly). This feature will not be required for applications where the oriented parts are to be presented directly to assembly stations; the approaches defined for this application are not restricted to palletized parts only.
2. Individual inspection/palletizing cells will be used for each part group, with some combining of part groups if practical. However, a uniform system architecture will be supported across all inspection/palletizing cells to maximize interchangeability, simplify maintenance, and minimize development effort.
3. Generalized singulating, selection, and orienting approaches will be used where applicable (to support system modularity, reconfigurability, and minimal implementation cost).
4. 100% inspection will be provided for only first level features (due to high number of details for full inspection plus the high production rate).
5. Partial sampling with integral automated SPC for the full set of part dimensions will be supported by a reconfigurable Vision Gauging Module.

6. The inspection/palletizing cell will be controllable either manually (through a touch screen Man-Machine Interface) or automatically (via the robot/vision controller).

7. The VGM (and to a lesser degree, the orienting station) will utilize uniform mechanical components, electrical components, optics, and software across all inspection/palletizing cells.

Figure 2 presents an overview schematic of the Modular Inspection/Palletization Cell approach. As shown, the Vision Gauging Module can be sited directly adjacent to the Orienting/Palletizing station, where it can be loaded/unloaded and controlled by the orienting/palletizing robot. The VGM can also be sited remotely from the Orienting/Palletizing station where it can be loaded/unloaded and controlled manually. This is required for gauging parts at various upstream points in the fabrication process and also for gauging parts supplied from outside the plant.

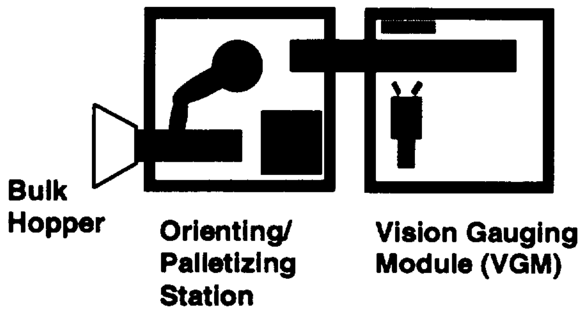


Figure 2 - The Modular Inspection/Palletization Cell Concept

7.0 The Prototype Vision Gauging Module

The Vision Gauging Module incorporates a number of the MARAS attributes noted above. Since the VGM is further developed than the Adapatable Feeding Module or AFM, it will be used to further illustrate these concepts.

Consistent with the core MARAS concept, the following features were included as part of the prototype VGM design:

1. Modular building block elements (camera mounts, light source mounts, calibration targets, electrical components, etc.).

2. Use of a generalized nest block to hold parts in various orientations before optics.

3. Design for loading by hand, robot, or fixed automation.

4. Reconfigurable software (via setup file and downloaded set points and recipes).

7.1 The VGM Station Base

Figure 3 presents the general purpose nature of the station base employed for the VGM. Extruded aluminum profile sections were used to fabricate the frame and table base for mounting the optics, electronics, traversing slide, and other components. The table base is formed from adjacent 160 mm x 40 mm extruded sections, providing a mounting base very similar to the T-slot type of base often employed for machining fixture mounting. A 0.75 meter (30 inch) servo-controlled traversing rail slide is mounted to the top of the table base to index parts to be gauged before the appropriate optics.

The VGM station table base can support mounting of up to three to four cameras or mechanical gauging subsystems plus required illumination sources (front lighting, back lighting, structured lighting, etc.). A family of general purpose mounts has been developed to address quick and flexible placement of these gauging elements or "gaugels" on the VGM table base. This facilitates efficient setup or changeover if entirely new lighting approaches and/or lens characteristics are required for a new gauging application

The NEMA 12 enclosure mounted to the lower side of the table base provides a sealed and air-conditioned housing for the supervisory control computer, network interfaces, illumination sources, power supplies, and input/output subsystems. Sliding opaque door panels (not shown) are installed between the frame sections above the table base. These are to minimize dust infiltration and background lighting disturbances in the gauging area. The touch screen operator interface panel is installed to the top of the VGM station base frame on a swivel base. The screen centerline is at 1.7 meters above floor level for optimum ergonometrics.

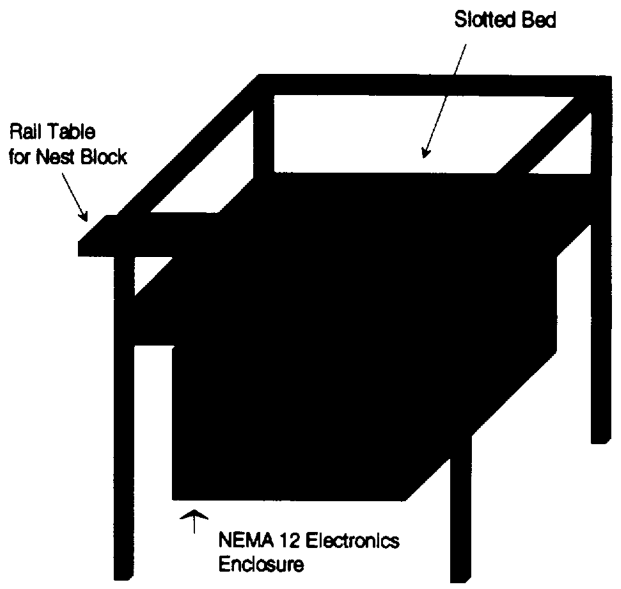


Figure 3 - Vision Gauging Module Station Base

7.2 The VGM Generalized Nest Block Approach

The VGM must be able to support easily reconfigurable fixturing or gripping approaches to facilitate a wide range of part geometries and gauging functions in an adaptable manner. A generalized nest block fixture design is employed for this purpose, as shown schematically in Figure 4. Here, the nest block base is a CNC-produced fixture block that includes aligning nests for holding parts in the orientations required for vision gauging all the required dimensions and features. Parts are placed approximately in the nest from above by hand or by automated means. Next, a mating CNC-produced clamping "plate" is actuated by either operator or automatic command to provide final and deterministic positioning of the parts in the nest block. Although the vision system can compensate for positional inaccuracies perpendicular to the view axis, accurate and repeatable alignment is especially important in the direction along the view axis to maintain focus integrity when the field of view (and thus, depth of field) is small.

Proximity sensors mounted to the nest block are used to verify clamp bar activation and deactivation. An integral calibration target is incorporated into the nest block for ease of optics alignment and calibration.
Although simple geometries are shown in Figure 4, a typical nest block and clamp bar include numerous details to provide the required aligning/orienting functions plus optical paths to view the required part sections. Using advanced CAD systems and CNC greatly streamlines the design and fabrication of the nest block and clamp bar for any required part.

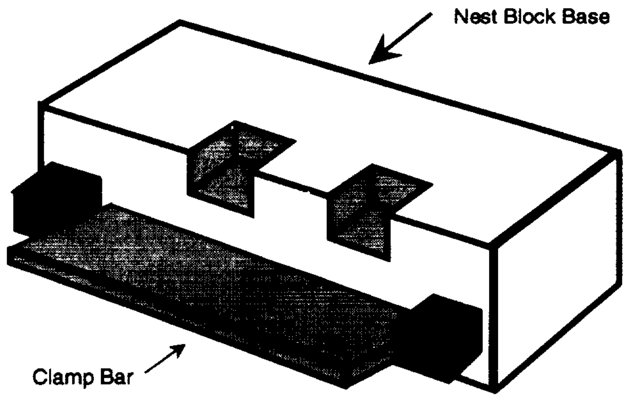


Figure 4 - Vision Gauging Module Generalized Nest Block

7.3 Vision Gauging Module Elements Representation

A key component of the MARAS concept is the representation of the elemental building blocks. This includes physical elements as well as control definitions. The representation of an object typically includes geometric definitions plus extended parameters that describe other attributes such as ranges of control or actuation supported, discrete operating states, mating requirements for integration with other modules, etc.

For the prototype VGM systems, Table 4 provides a sample of typical element definitions:

Table 4

Sample Object Parameters or Properties

Object	Sample Parameters
Back light module	Size of illumination area Light intensity range Available mounting elements Available illuminator elements
Laser line source	Range of focal lengths Ratio of line width to length Available mounting elements Available light intensities
Nest block	Focal centerline height Number of nests Number of actuators, sensors Actuator and sensor connects
Camera lens	Focal length range FOV range
Gauging shot	Illumination modules used Associated gauge functions Number of analysis steps Nest translation for shot Calibration set parameters

Although the examples presented in Table 4 are by no means a complete set of the object definitions used for the prototype VGM, it does illustrate the type of information contained within these definitions.

7.4 Inspecting/Palletizing Cell Control Architecture

Figure 5 provides a block diagram view of the major components employed for the Inspection/Palletizing cells. This includes an Intel 486 based supervisory computer running Microsoft Windows that functions as the supervisory controller, man-machine interface, SPC analyst, and production tracking system for each inspection/palletizing cell. These supervisory computers are also networked together via ethernet to support upload of summary information to higher level plant systems. Remote access to these systems is supported via modem communications for service diagnostic and maintenance purposes.

The software used to operate the supervisory computer is a next generation derivative of VAX/VMS and OS/2 based factory control software systems originally implemented for discrete parts assembly/test at facilities such as Chrysler, General Motors, and Caterpillar in the late 1980s. This software was entirely re-written and enhanced over the last year utilizing current object-oriented development tools and coding approaches.

Additional information regarding the supervisory computer software functions is presented in section 7.5 below.

Emulation of Modicon Modbus communications protocols is an important feature of this architecture. This provides the flexibility to communicate to a wide variety of robotic/vision controllers and other intelligent devices via relatively simple serial line connections. This communications link is used to collect operating status and process data from the robot/vision controller for the orienting/palletizing station and one or more VGM stations. Process setpoints, recipes, and status change commands (such as clamp station, start station, stop station, abort cycle, etc.) are also downloaded to the robot/vision controller via the emulated Modbus. In addition, the Modbus protocol was extended to support automated download of setup or configuration files to the robot/vision controller over the same line used for production data uploads and command/recipe downloads.

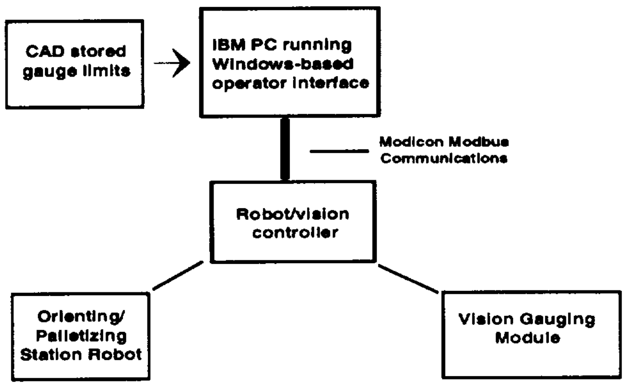

CAD stored gauge limits
IBM PC running Windows-based operator interface
Modicon Modbus Communications
Robot/vision controller
Orienting/ Palletizing Station Robot
Vision Gauging Module

Figure 5 - Inspection/Palletization Cell Control Components Schematic

The software developed for the robot/vision controller is another vital component of the overall architecture. Based on commands, setpoints, and setup file information downloaded from the supervisory controller, the main software control program in the robot/vision controller performs the following functions:

1. Controls light source activation and nest block clamping activation (via Opto 22 Optomux network modules).
2. Senses states of nest block clamp (via proximity switches linked to Opto 22 Optomux network modules).
3. Commands traversing slide to index nest block to required positions.
4. Performs setup file defined vision gauging functions (frame acquisition, vision algorithm execution, numeric functions, data analysis functions).
5. Updates status block with most recent gauge results and process information for collection by the supervisory computer.

All of these functions are definable by the setup file downloaded from the supervisory controller. The setup file contains sets of parameters which completely define the required functions of the VGM. This provides the capability of downloading a setup change "on-the-fly" without the need to halt non-affected operations. The setup file is much more compact than downloading new programs to the robot/vision controller. Thus, the time required to implement a given VGM setup change is short. A set of setup files can be maintained on the supervisory computer in a protected directory that can only be accessed by authorized plant personnel. Currently, the files are maintained manually by used of a text editor.

The setup files define such parameters and selectable features as:

1. Number of cameras defined for the VGM.
2. Illumination sources to be used.
3. Nest block positions for each vision frame (camera shot) to be acquired.
4. The specific vision and numerical algorithms to be employed (including sequencing and execution parameters) for each camera shot.
5. Number of gauge variables to be tested, including process limits.
6. Association of gauging functions with processing stations or operations (for SPC purposes).
7. Reject condition codes.
8. Auto detection of optics failure.
9. Required clamping confirmation input identification (if any).
10. Error handling functions.

Setup file definitions can also be model (part type) specific. That is to say, a set of parameters defined in the setup file can be associated with a particular "model" or part variation. The setup file can thus define the operation of the VGM to be unique for different part variations. With appropriate definition of the setup file, the VGM will automatically adapt itself to process different parts via simple download of a new model code from the supervisory computer.

7.5 The VGM Supervisory Control Computer Software

The supervisory computer software must be very intuitive and easy to use for effective operation by plant floor personnel. Appropriate use of graphical user interface elements has thus been used towards this end in implementing the software.

A menu bar at the top of the screen has been kept purposefully simple for ease of use when performing common operations. Toolbar command button icons have also been employed to provide quick access to the most frequent functions, such as:

1. Starting and stopping the system monitoring functions (password protected).
2. Display of communications status.
3. Station selection for additional detail
4. Display of station control panel..
5. Display of station alarm/fault and production counts summary status.
6. Display of defined CAD images.

7. Display of raw process data from monitored stations.
8. Access to historical production trend displays.
9. Access to historical alarm/fault trend displays.
10. Access to SPC tracking displays, logs, and charts.
11. Access to the on-line help system.

8.0 Orienting and Singulating Concepts for Flat Parts

This section presents some of the basic general feeding approaches we have been pursuing as part of our preliminary definitions of an Adaptable Feeding Module (AFM). We are using these plus more complex approaches to orienting using both manipulators and passive handlers (using the terminology of Boothroyd, Poli, and Murch[11]) to define our first set of "feedel" elements plus corresponding parameter set definitions and control software objects.

Given the wider range of potential future applicability to other part families, we have first concentrated on flat parts to define candidate generalized singulating approaches. Singulation is the process of separating parts into a single vertical layer with only one or possibly two orientations ("top" up or "top" down).

For flat parts singulation, the following attributes are common:

1. Parts are originally in a bulk bin or hopper that dumps into a vibratory hopper (not bowl feeder) that dispenses a steady stream of parts onto a conveyor.

2. All parts don't need to be oriented. Allowing for de-selection of some parts to recirculate can make for a more robust and practical system design that also naturally supports part purging for changeover.

3. Generalized end-of-arm-tooling should be used where applicable (such as vacuum cups for flat parts).

4. Use generalized and/or modular singulating elements or bars.

5. Use vision where aligning/orienting is not possible or practical by geometric means only.

6. Use common mechanical components and control software for each part type.

By definition, we refer to a flat part as one where the thickness of one dominating geometric surface or plane of the part to be fed or oriented is much less the height and width of the surface. Examples include parts stamped from sheet metal where the resulting flat feature surface dimensions are large compared to the thickness of the part. For the target application of cylinder lock assembly, the keys fall nicely into this category. Some members of the driver part family may also apply. Given the ratio of the cap height to cap diameter of approximately 0.3, this should apply to caps as well.

Figure 6 provides a simple overview of one approach to flat parts singulation. Here, the vibrating hopper feeds a conveyor that indexes parts towards an area where a machine vision camera is used to verify part position and orientation for acquisition by a robotic gripper. Wiper blades over the conveyor are used to achieve a single layer of parts. Narrowing blocks over the conveyor confine the parts to a specific region for the first level vision inspection and robotic acquisition for final orienting. This method is quite effective and can supply a steady stream of singulated parts. However, parts sometimes jam. This makes the approach unreliable. The use of a rotating wiper or brush can potentially alleviate this problem.

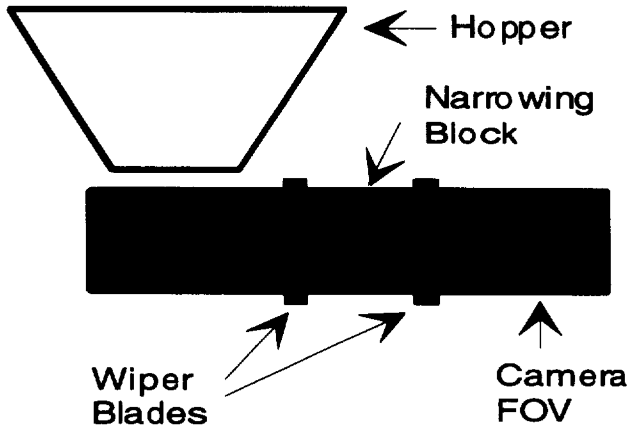


Figure 6 - Singulation of Flat Parts with Wiper Blades and Narrowing Blocks

Figure 7 presents another potential approach for flat parts singulation. Here, the hopper feeds a singulating ramp with shelves that deposit a single layer of parts on the conveyor. The ramp is sloped down towards the camera FOV and also away from the hopper. The lip height of each shelf is equal to the height of the flat part. Thus, parts will either slide off to the overflow area of the conveyor or fall into one of the shelves and slide down the shelf to the conveyor to be advanced to the camera FOV.

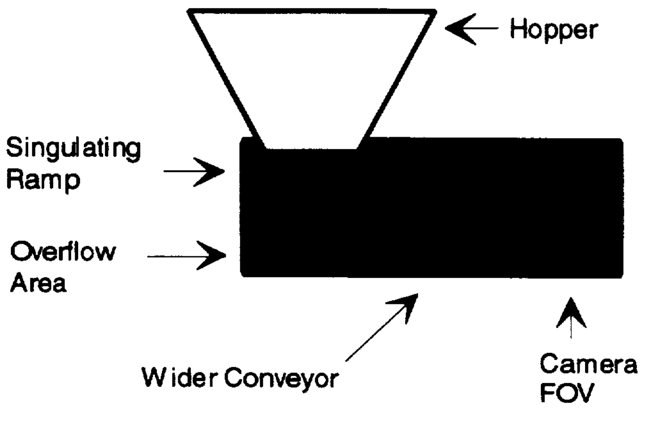


Figure 7 - Singulation of Flat Parts with a Singulating Ramp

Problems with this approach include a high ratio of overflow parts versus singulated parts and a less steady flow of singulated parts to the camera FOV.

9.0 Future Work

These two sample approaches are not intended to imply the full range of options available for singulating or orienting parts. However, they do serve to illustrate some fundamental principles worth considering in defining or implementing a generalized orienting system.

These and other approaches are being refined and verified for application to the cylinder lock application and other small mechanical part assemblies. Common to each of these approaches is the need for a modular and reconfigurable architecture in both the physical and software components. This applies to the guiding or aligning elements, vision systems, and the additional orienting functions performed by some sort of robotic gripper.

For the near term, the immediate goal is to complete installation of the first three inspection/palletization cells in the first quarter of 1994. Although these first installations will incorporate some of the reconfigurable features of the VGM for their feeding and orienting functions, it is expected that this will be even more so for the next two inspection/palletization cells to be completed later in 1994.

Application of these principles for adaptive gauging and feeding is now in progress for three other automated assembly projects to be completed towards the end of 1994. Additional integration of CAD modelling for automated or semi-automated synthesis of appropriate adaptable system configurations is planned.

References

1. Andreasen, M. M., Kahler, S., and Lund, T., *Design for Assembly*, IFS Publications Ltd., U.K., 1983.

2. Natarajan, B. K., "Some Paradigms for the Automated Design of Parts Feeders", *The International Journal of Robotics Research*, vol. 8, no. 6, pp 98-109, December 1989.

3. Cohen, G., "Simulated Intelligent Flexible Cell for the Assembly of Multi-Component Systems", *Engineering Applications of Artificial Intelligence*, vol. 4, no. 2, pp 121-130, 1991.

4. DaCosta, F., et al, "An Integrated Prototyping Environment for programmable Automation", *International Symposium on Intelligent Robots in Space*, (SPIE/OE 1992).

5. Womack, J. P., Jones, D. T., and Roos, D., *The Machine that Changed the World*, Macmillan Publishing Company, 1990.

6. Wheelwright, S. and Clark, K., *Revolutionizing Product Development*, The Free Press, 1992.

7. Asada, H. and By, A., "Implementing Automatic Setup Change Via Robots to Achieve Adaptable Assembly", Proceedings of the 1984 American Control Conference, 1984.

8. Asada, H. and By, A., "Kinematic Analysis of Workpart Fixturing for Flexible Assembly with Automatically Reconfigurable Fixtures", *IEEE Journal of Robotics and Automation*, RA-1 (No. 2), 1985, pp. 86-94.

9. Scheinman, V., "ROBOTWORLD, A Multiple Robot Vision Guided Assembly System", *Robotics Research, The Fourth International Symposium*, The MIT Press, 1988.

10. Khosla, P, Kanade, T., Hoffman, R., Schmitz, D., and Delouis, M., "The Carnegie Mellon Reconfigurable Modular Manipulator System Project", *Robotics Institute Research Review*, Robotics Institute, Carnegie Mellon University, 1988.

11. Boothroyd, G., Poli, C., and Murch, L. E., *Automatic Assembly*, Marcel Decker, New York, 1982.

N/5 -

207631

p- 4

6029

AIAA-94-1184-CP

N94- 30534

AGILE MANUFACTURING - THE FACTORY OF THE FUTURE

Joseph M. Loibl
Terry A. Bossieux
Ford Motor Company
Alpha Simultaneous Engineering
AMTAC
15100 Mercantile Drive
Dearborn, Michigan 48120

Abstract

The Factory Of The Future will require an operating methodology which effectively utilizes all of the elements of product design, manufacturing and delivery. The process must respond rapidly to changes in product demand, product mix, design changes or changes in the raw materials. To achieve agility in a manufacturing operation, the design and development of the manufacturing processes must focus on customer satisfaction. Achieving greatest results requires that the manufacturing process be considered from product concept through sales. This provides the best opportunity to built a quality product for the customer at a reasonable price.

The primary elements of a manufacturing system include people, equipment, materials, methods and the environment. The most significant and most agile element in any process is the human resource. Only with a highly trained, knowledgeable work force can the proper methods be applied to efficiently process materials with machinery which is predictable, reliable and flexible.

This paper discusses the affect of each element on the development of agile manufacturing system.

Introduction

To be competitive in the world market an organization must efficiently utilize all of its assets. The traditional elements of the manufacturing process are men, machines and materials which are combined using proven and consistent methods which are responsive to a rapidly changing environment. (Figure 1). The manufacturing

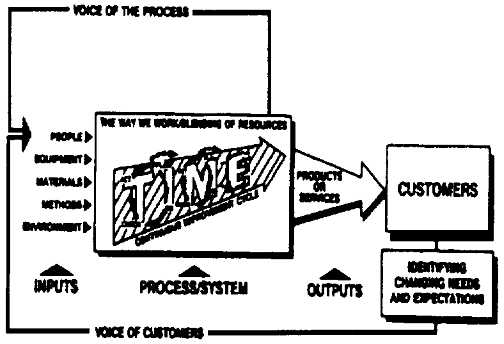


FIGURE 1 - PRODUCTION CYCLE

system must be capable of producing the right products, in the needed quantities with high quality and the lowest possible cost. An agile manufacturing process can only be achieved if the associated processes are designed concurrently with the product utilizing a crossfunctional, simultaneous engineering team comprised of representatives of all affected organizations. The manufacturing concept is revised as the team proceeds through the product development cycle as shown in Figure 2. This defines the various stages of the product life cycle.

CONCEPT
DESIGN
PROTOTYPE
DEVELOPMENT
PRODUCTION
MARKETING
SALES
SERVICE
REUSE/RECYCLE

FIGURE 2 - PRODUCT DEVELOPMENT CYCLE

Flexibility starts with the design of the product. Use of techniques such as Design for Manufacture(DFM), Design for Assembly(DFA), Quality Function Deployment(QFD), Statistical Process Control(SPC), Design Of Experiments (DOE) and Computer Aids such as CAD/CAM/CAE, including product and process simulation, will be essential to develop a system which can respond rapidly to product changes, product changeovers and variation in the product mix. The overall production system will only be as agile as the least agile of the elements of the system.

Flexibility must exist in the product design, the process design, the production system and the material handling operations. An agile organization will allow the operations to respond to the needs of the customer as demanded by the ever changing market in the shortest amount of time. This includes the capability to alter the mix among• several similar products within the manufacturing capacity (i.e. volume mix flexibility) as well as the ability to rapidly convert to new products which utilize common manufacturing equipment (i.e. product changeover flexibility).

An agile operation can only be achieved if this objective is considered from the conception of the product through sales and service. Agility must be a major objective of the development and must be planned and built into the process. During the development process, a simultaneous engineering approach is used which considers the capabilities of the process as well as the needs of the product to meet customer expectations. Each element in the product equation, Men, Machines, Materials, Methods and the Environment, is evaluated and optimized. When tradeoffs are considered, the decisions are based on providing the best value for the customer.

Human Resources

The most flexible component in the process is the human resource. Important characteristics of the Human Resources are shown in Figure 3. It will be essential that all employees are highly trained individuals who are knowledgeable about the latest technologies and specifically trained in the equipment that they use on a day to day basis. They

FLEXIBLE HUMAN RESOURCES
- **ENGINEERS**
- **MANAGEMENT**
- **PRODUCTION**
- **SUPERVISION**
- **SERVICE ORGANIZATIONS**

CUSTOMER ORIENTED
HIGHLY TRAINED
MOTIVATED
TECHNOLOGY KNOWLEDGEABLE
CONTINUOUS SKILLS IMPROVEMENT

FIGURE 3 - HUMAN RESOURCES

will also need to maintain their skills through a continuous personal enrichment program. The workers of today will not be competitive in the environment of the agile manufacturing system without a comprehensive and effective plan to maintain and enhance employee skills.

The successful enterprise will encourage employees to continuously improve their skills by providing opportunities to attend training related to specific job requirements. Their effectiveness will depend on the company's ability to provide incentives for the associates which assure continuous improvement in the abilities of all employees. It is necessary to have an educated, flexible, empowered and motivated work force to respond to the needs of the customer.

Equipment

Another factor of production is the machinery or equipment which is used to build a product or provide a service. The equipment although an important component of the process is limited by the ingenuity of the people who design, develop and operate it. The equipment is a minor albeit essential part of the overall system. The primary characteristics of the machinery is the ability to reuse or reapply the equipment to respond to variations in product mix and to provide sufficient flexibility to be used with new products.

Other features of the Factory of the Future include reliability, maintainability and the ability to rapidly redeploy equipment. See Figure 4 for a list of the important equipment characteristics.

Flexible equipment such as robots, AGVs, ASRSs, CNC machines, programmable controllers, personal computers, modular conveyors, coordinate measuring machines, smart instruments and intelligent sensors are all important for the agile manufacturing system.

FLEXIBLE
- **Multi-Use**
- **Multi-Product**
- **Rapidly Reconfigurable**
- **Product Mix**

REUSABLE
- **Rapidly Change To Different Parts**

RELOCATABLE

PROGRAMMABLE/REPROGRAMMABLE
- **Off Line**

EXPANSIBILITY (Capacity)

RELIABLE

MAINTAINABLE

FIGURE 4 EQUIPMENT (MACHINERY)

Materials

The materials of production can refer to product components or materials of the manufacturing process. Alternative materials are evaluated throughout the product development cycle to consider the physical and chemical property requirements and select the materials which provide the most cost effective option for both product and process equipment. Figure 5 identifies some the situations where consideration of materials is important.

Effective selection of materials can have a significant affect on the life cycle cost of a manufacturing process with associated influence on the cost of the product(s).

Materials used in products, tooling and process equipment are each important in their own way. The physical and mechanical properties of the materials affect the life of components, durability, reusability and recyclability.

TOOLING
- **Reusable**
- **Recyclable**

NEW RAW MATERIALS
- **Steel vs. Aluminum vs. Magnesium vs. Composites**
- **Plastics**
- **Thermoplastics vs. Thermosets**

RAPID PROTOTYPING
- **Stereolithography**
- **Cubital**

FIGURE 5 MATERIALS

In the design and purchase of equipment and tooling, it is important to consider how it might be used to process parts for several optional materials. As an example, a painting process has similar requirements for capacity and capability regardless of the material applied or the substrate. However, the properties of the coating may change which in turn requires a change to the process parameters. Different types of nozzles, paints guns or controls may be utilized while maintaining the same basic system. This flexibility may be required to adjust for viscosity variation in the material as well as different curing requirements. The paint process must be robust in the ability to produce a quality paint job using many different paint combinations and accommodate changes in environmental conditions. This must be accomplished with little or no change to the base equipment and the necessary changes must be easily implemented.

Materials used in the tooling and equipment are also evaluated to determine the most effective use of specialty compounds. In the ideal situation the tooling components will wear out just as the product cycle is complete.

Rapid prototyping is an emerging technology which enables the preparation of prototype parts much faster than available from previous practice. Methods such as stereolithography, cubital and other similar techniques utilize special chemical and physical properties of materials to effectively reduce the time required to produce prototype parts. In some instances this time has been reduced from months to weeks. The processes achieve these dramatic improvements by operating directly from CAD data. The CAD data is used to initiate these processes. The data is used to operate a numerically controlled device which automatically replicates the part design. This bypasses time consuming manual design detailing and the machining and build up of the parts.

Selection of materials during every stage of the development is important. The material choice affects the product cost and quality and may also influence the time to produce parts.

<u>Methods</u>

With the exception of the human resources, the most influential factor of the agile manufacturing organization relates to the methods which are implemented through out all phases of the product development cycle.

Frequently, the design, development and operating methods are the elements which define the agility and flexibility of a system. This may be a an equipment operating procedure, the control system, an accounting procedure or a management practice. Consideration is given to all aspects of the business enterprise if all components are to work effectively together. Some effective methods of improving communications and intra-organizational cooperation are identified in Table 1.

One process which has achieved significant improvements in total development time, reductions in cost and improvements in quality is the crossfunctional team. Combined with simultaneous engineering of the product and process, significant benefits are realizable in the manufacture of a product. The crossfuntional team involves representatives from design, product engineering, manufacturing engineering, production and suppliers. During the entire product design and development cycle, the team uses many of the computer based tools, e.g. CAD, CAM, CAE and CIM, and statistical methods to accelerate the design and development process. The use of computer tools is an essential element in the process but it is the knowledge and ability of the human resources which is necessary for the effective implementation of these tools.

These techniques provide significant benefits during the early phases of the development process. These improvements must also be carried to the plant floor to achieve the flexibility in the manufacturing process. This is achieved through user-friendly operator interface which can be used in the setup and control of the manufacturing equipment.

The manufacturing process is designed in cooperation with product design, engineering and production. With this approach, the resulting product design is robust with regard to manufacturing capability. With a focus on manufacturing flexibility a more agile manufacturing system is the result.

As listed in the Table, there are many other procedures and methods which are used to improve the development system. Procedures for the selection and justification of equipment can significantly affect the ultimate decision. Focus on the traditional Return on Investment(ROI) often leads to decisions which are not compatible with the agile manufacturing needs. New methods which consider life cycle cost, the cost of quality and activity based accounting provide consideration of the value of some of the intangibles in the equipment purchase decision.

<u>Environment</u>

In addition to the four factors previously discussed, the process must be responsive to changes in the environment in which it operates. This must be accomplished rapidly to maintain the agility of the system. Figure 6 identifies some of the important environmental or external factors which may affect the process. There are numerous external factors which can be considered. These may have a significant affect on the organization depending on its particular business.

We have seen the substantive influence that government regulations and policy can have on the operation of an enterprise. In addition local work practices, internal standards, accepted national/international codes and standards and changes in the global situation affect the operating

GOVERNMENT
- **OSHA**
- **EPA**
- **Tax Regulations**
- **Safety Standards**
- **Labor Regulations**
- **Government Subsidies**
- **ADA**

SOCIAL RESPONSIBILITY

CHANGES IN THE GLOBAL SITUATIONS
- **Political**
- **Economic**
- **Trade Agreements**

WORK PRACTICES

STANDARDS AND PRACTICES
- **Engineering**
  ANSI, RIA, ASTM, AIAG IEEE
- **Financial & Accounting**
- **National Codes (e.g. UL)**

FIGURE 6. ENVIRONMENTAL

efficiency and the competitive position of a business.

Changes in the environmental factors can result in rapid and dramatic changes in the factors of production, human resources, materials, equipment and business methods. For example, changes in the standards implemented by a country or group of countries, affects the ability to sell products in certain markets or can cause a change in the availability of certain commodities without any other local changes in the operations.

Likewise, political changes may influence the competitive position quickly and dramatically.

Sometimes, changes in the environment can be anticipated but very seldom can they be controlled. Many of these changes, especially those which are the result of legislation, occur over a long period of time. Plans can be implemented to adjust for these changes. However, in other situations, political or governmental changes may be rapid and cataclysmic. In the latter case, a rapid response is required to maintain competitive position. This can only be accomplished by an enterprise which is designed and developed to support agility in the operations.

Conclusions

In this ever changing world, only the strong and the agile will survive. To be a successful organization, the agile business enterprise will focus on the ability to rapidly respond to customer need and provide quality parts at a price that represents value to the customer. This requires that all of the factors of production are developed with flexibility and agility in mind. This must commence with the product concept and carry through to the sale and marketing of the product.

METHOD	CON- CEPT	DE- SIGN	PROTO TYPE	DEVEL OPMEN T	PRO DUC TION	MAR KET ING	SALES	SER VICE
MARKET STUDIES	*	*				*	*	*
CUSTOMER INPUT	*	*				*	*	*
SURVEYS	*	*				*	*	*
DFM		*	*	*	*			
DFA		*	*	*	*			
SIMULTANEOUS ENGRG.	*	*	*	*	*			*
SUPPLIER INPUT	*				*	*	*	*
DFE	*				*	*	*	*
UPFRONT ENGRG	*	*	*	*	*	*	*	*
CAM		*	*	*	*			
CAE	*	*	*	*	*			
DESIGN FOR QUALITY		*	*	*	*			
SIMULATION	*	*		*	*			
OLP				*	*			
STAT METHODS		*	*	*	*	*	*	*
SCHEDULING				*	*		*	
ORDER ENTRY					*		*	
PRODUCTION CONTROL					*			
INVENTORY CONTROL					*		*	*
PURCHASING					*			
DOE				*	*			
PM					*			
SPC					*			
NEURAL NETS				*	*			
PRODUCTION MONITOR					*			
OPERATING PROCEDURE		*	*	*	*			
CONTROLS					*			
HUMAN INTERFACE		*			*			
RAPID PROTOTYPE		*	*	*				
ORDER TRACKING					*		*	
DELIVERY					*		*	*
FOLLOW-UP	*					*	*	
MAINTAINABILITY					*			
CONT. IMPROVEMENT	*	*	*	*	*	*	*	*

6030

AIAA-94-1185-CP

N94- 30535

Confessions of a Robot Lobotomist

R. Marc Gottshall - Specialist Engineer
Boeing Commercial Airplane Group
Seattle, Washington

p. 10

Abstract

Since its inception, Numerically Controlled (NC) machining methods have been used throughout the aerospace industry to mill, drill, and turn complex shapes by sequentially stepping through motion programs. However, the recent demand for more precision, faster feeds, exotic sensors, and branching execution have existing Computer Numerical Control (CNC) and Distributed Numerical Control (DNC) systems running at maximum controller capacity. Typical disadvantages of current CNC's include fixed memory capacities, limited communication ports, and the use of multiple control languages. The need to tailor CNC's to meet specific applications, whether it be expanded memory, additional communications, or integrated vision, often requires replacing the original controller supplied with the commercial machine tool with a more powerful and capable system.

This paper briefly describes the process and equipment requirements for new controllers and their evolutionary implementation in an aerospace environment. The process of controller retrofit with currently available machines is examined, along with several case studies and their computational and architectural implications.

Introduction

In response to the more complex machined shapes demanded by modern aircraft, the Air Force sponsored numerically controlled milling machine research at the Massachusetts Institute of Technology's Radiation Laboratory in 1949. The fusion of the then fledgling digital computer technology with servo control techniques allowed demonstration of a prototype NC machine in 1953 [1]. Over the ensuing forty years, new CNC capabilities have dramatically enhanced the way airplanes are made. CNC computers have become smaller, faster, and cheaper; through the use of innovative sensors, automated work cells can both monitor and control production processes as well as the parts they create. Upstream systems can create and store part programs, collect and analyze process data, and monitor/diagnose individual machines. In general, the processes being performed are more complex, highly precise, intolerant of delay, and are being automated at an ever-accelerated pace.

When tailoring a controller for a machine tool application, two critical considerations must be taken into account: process complexity and life cycle cost. The desire to improve product quality and reduce manual labor has caused automated systems to become more and more sophisticated. On the control side, automation applications require ever increasing amounts of software that execute on powerful computers with extensive memory. On the process side, smart-sensor based systems provide tighter control of production monitoring, quality, and reliability by collecting massive amounts of data during process execution. This data must be organized for use by both the process control and upstream business systems. Clearly, what was once a single computer operation has now become a network of 5 to 10 intelligent computer subsystems, each of which is usually a microprocessor-based smart box. The function of each subsystem is unique yet all subsystems contribute to producing a better product.

Examples of smart-sensor based subsystems include machine vision for process inspection and statistical analysis, and thermal scanning devices to monitor material growth. Data transfer of part attributes, quantities, and messages require networking capability to disk storage, file management , and company business systems. Further complicating the automation process is the need for a host system which is flexible enough to coordinate all subsystem information and make adjustments to the process in real-time. The host must also interface hardware and software to multiple communication protocols.

Cost and Complexity

While issues regarding process complexity represent the factory side of the automation problem, the business side is concerned with controlling cost. The vast amounts of software generated for application development, programming, and software maintenance must be structured in order to control life cycle costs. Because these automation systems are multi-computer based, organizing and directing in-process information mandates complex decision making algorithms. For example, many processes require the system to adapt to changes in the process based upon input data, factory problems, and machine interrupts. To effectively implement such complex process algorithms, application software is usually developed using structured analysis and design. Structure design tools benefit the software life cycle in development, maintenance, and documentation. However, it is not always possible to take advantage of cost savings using structured design tools unless the computer language can support such development.

Typical software maintenance costs for complex automation applications can be excessive due to the diversity of languages, controllers, and variety of processes. For example, most NC, CNC, and DNC machines utilize control language based upon ladder logic. Other languages such as Allen Bradley's Siprom are used in conjunction with ladder logic when developing a machine application. Large multi-function systems written entirely in ladder logic pose a formidable maintenance task . The maintenance problem is compounded further since robotic control systems often use custom languages (such as Karel, Rail, V+, etc.). Each of these unique languages must be supported by programming staff. Factors such as language, processing capability, interconnectivity, communications, and code reusability must be weighed against what the company can afford to spend throughout the software life cycle.

The issues of process complexity and software life cycle are interdependent in the automation environment. The interdependence can be examined by breaking down these issues into further detail. First, process complexity involves key factors such as programming, communications, data transfer, control of input/output functions, and motion control. Life cycle costs, on the other hand, involve computer languages, maintenance, training, upstream compatibility, and software reusability.

Process software can be partitioned into six distinct functional groups. Generalized categories include process control, communications, file storage and transfer, digital and analog input/output, motion control, and vision processing. Of these categories, serial communications has become a critical link for most automation applications within Boeing.

Serial Communications

Many new applications utilize microprocessor-based smart boxes which can control an entire section of a process with little intervention from a host computer. The ability to allocate tasks to multiple smart boxes reduces the work load on the main controller. In addition, it provides system modularity which can reduce factory down time and part replacement. The majority of these smart boxes provide serial ports for communication. In order to reliably communicate with multiple smart boxes, the system programmer needs to have standard serial communication functions available within the host controller's language. A set of common tools might include full ASCII character recognition, basic character input/output, and configuration of the I/O port. Advanced features include data buffering, operating system notification (via flags or interrupts), and the ability to apply protocols such as Kermit, Xmodem, etc. to data transfer. Many controllers do not allow much control over a serial port, resulting in "kludging" the existing software base to create a semi-functional communications path.

Several aerospace applications require the use of thermal scanners for monitoring temperature changes and part growth the work cell. Interfacing and manipulating the data provided from these scanners has proven to be a programming challenge. Each controller has a unique implementation of the RS-232 standard. Furthermore, some controllers use restricted data formats, which limit the flexibility of the system. Still other controllers require special manipulation of the serial port hardware to make the port functional. Consequently, special communications software must be written after the serial port has been studied through a network analyzer. Compounding the problem is the lack of an RS-232 standard on the smart device. The result is the communications software must not only conform to a non-standard format at the controller side but also on the sensor side.

Protocols such as Kermit, Xmodem etc. have been successfully used in the computer industry for years. As more embedded PC boxes sprout up in automation applications, the need for a robust communications tool set resident in both the host controller and sensor systems is continually overlooked. In addition to serial communications, smart boxes are synchronizing communication with digital I/O. End effectors and manual operator interfaces can use combinations of serial communications, discrete digital I/O, and analog input/output. End effectors can be considered as completely independent machine processes. Smart controllers are used with end effectors to control valves, drill motors and part manipulators. Here again, serial communication is used to set up the end effector and control the process in real time.

Digital I/O Control

Assembly and manufacturing applications require synchronization of multiple control relays and valves using discrete digital I/O. Process control is dependent upon the ability of a host controller to receive serial information and/or discrete digital I/O, decode the information, then make a decision affecting the next step in the process. Programmable Logic Controllers (PLC's) have been used for this task. The PLC is a cornerstone in many Boeing automation applications due to its "bulletproof" ability to control process I/O. Other benefits include a large base of people who program and trouble shoot in ladder logic.

In addition to PLC's, most control system manufacturers provide both digital and analog I/O. These I/O's are interfaced to operator control panels, process switches, valves, and a multitude of sensors and indicators. While I/O interfacing is somewhat standardized, tools for developing I/O control algorithms are not. Programming a PLC for interfacing to an operator control panel can be difficult due to the lack of a rich language base. Designing a system in which I/O's can be placed in logical groups is dependent on where the grouping takes place and how many I/O's are required.

Distributed I/O boxes aid in modularizing the system design, but also complicate the system by the sheer numbers of sensors being processed. The host controller must have intelligent control over all I/O's both in hardware and software. Many real-time processes require high speed processing of sensors in order to avoid catastrophic failure. This implies a group of dedicated high speed I/O's in addition to simple valve and switch control. The inherent nature of high speed data acquisition demands computing power as well as robust hardware. The problem is further complicated by the diversity of cables and connectors required to interface the sensors.

The basic process of reading a digital input or setting a digital output is not complex. However, when that process must be carried out at high speeds, the physics of transmission lines cannot be ignored. Further, the host controller may have to read several sensors at once, perform numerical computations on the data, iterate a decision tree, and execute a reactionary function. Adding to the myriad of hardware interfaces are the variety of timing requirements for data acquisition. Coordination of the system I/O's together with the application complexity generate huge amounts of control software.

Machine Motion

In many aerospace automation applications, the issues discussed above are secondary to precise control of machine motion. Machine motion is generally executed in joint or world coordinate systems. The dominant trajectories for machine controllers are joint or linear interpolated motion. The end result is to cause the tool tip attached to the machine to perform the required movement. NC machines utilize RS274D code to perform these movements. This standard was developed in the 1950's, before the application of matrix algebra in motion control. Today, robotic controllers use forward and inverse kinematics to drive multi-axis machines. Inverse kinematics allow the controller to compute where the tool tip is with respect to the coordinate base of the machine. This function is not possible with most NC machines.

Manufacture of aerospace grade parts demands high positioning tolerances on the part of the machine. NC machines have been capable of this for years provided the part being machined is always fixed in a specific position in the tooling jig. The NC machine can probe the part and account for offsets in the X, Y, and Z axes but it cannot adjust for changes in yaw, pitch, and roll. Preparation and assembly of parts such as fuselage panels involve path motion and positioning along complex contours. (This type of operation requires machines with 5 to 6 axes of motion.)

An NC controller can be programmed for complex motion but cannot adaptively adjust during the process. This is because RS274D code being executed by the machine is spatially fixed to either the machine or the part reference frame. Thermal growth affects machining tolerances due to the large size of many aerospace parts. The part, the tooling fixture, and the machine bed are subject to different growth fluctuations due to the materials they are built from. The goal is to produce a part with very high machining tolerances yet an NC machine cannot fully adapt to the dynamic growth changes caused by thermal effects. Controlling motion using kinematics has a distinct advantage by being able to dynamically create new frames of reference.

The part program is spatially fixed but a robotic controller can establish an offset reference frame in world coordinates using probing techniques. This reference frame can be used to transform the original part data to fit the current orientation of the part and tooling jig. Other processes require drilling of holes normal to the part surface. The normal vector and position must be computed just prior to drilling the hole. Again, this is not possible without the use of kinematics to locate the tool tip relative to the part. These operations require more computing power from the machine controller as well as the ability to store and transfer data generated by establishing in-process reference frames.

Language and Compatibility

Transferal of process data leads into the area of company business systems. The issue of upstream compatibility relates to the machine controller communicating through an established network protocol to a company data base. Unfortunately, upstream communications is tightly intertwined with the language used by the machine controller. Some systems use a server type architecture for communicating to the company database. This allows greater flexibility when changes are made to the system but the machine controller must still provide process information to some other computer based system. The focus of the next section is what role the machine controller language plays in interfacing not only to a server system but more importantly to the application itself.

The computer language of a control system plays the executive role in "gluing" application subsystems together. The language must provide a rich set of functions including input/output, file management, mathematical, decision iteratives, and graphics. Another important feature of the controller language is its ability to reflect the language syntax as readable structured text. It is extremely beneficial to be able to define and name software variables using meaningful words. Moreover, the extent to which the language lends itself to structured analysis and design implementations has far reaching impacts on costs incurred during the software life cycle. Automation software development, modification, and maintenance is a costly process within the Boeing company.

Utilizing multiple languages for an application has several drawbacks. Many companies worldwide use ladder logic as the standard for developing, implementing, and debugging sequential steps in automation and machining applications. Although newer languages may be far simpler to understand, an enormous base of people trained in ladder logic already exists. Reeducating such a large and sometimes unwilling work force is an immeasurable task.

Manufacturing companies have significant investments in existing machinery. Coupled to the machinery are support staff to maintain, operate, and reprogram production applications. Training for most of these companies is not economical. In addition, the choice of which control system and which language to standardize on is continually evolving.

Standardization of a subset of languages for applications is nearly impossible. Each automation application has specific requirements. These requirements cannot always be met using one manufacturers control system. A new system which fits the application may be purchased. This usually means a new control language with a different set of operating attributes and characteristics. Programming for the application now requires a "learning curve" with the new language, thus adding to software life cycle costs.

The variety of control systems, PLC's, and motion control cards used within Boeing are tied directly to the number of languages requiring maintenance and support. Each manufacturer has the "best" language for their machinery. Thus, every machine has one or more programming "specialists" intimate with that machine's language. Many of these machines have restricted language functionality.

Aerospace assembly applications require changes and modifications to the software as improvements are made in the process. When a controller with restricted language and/or functionality is used, the controller manufacturer must supply any customized software routines. These unique software requirements can add as much as 50% to the cost of the controller. Another cost burden is the lack of reusability of process code.

A company may expend considerable sums on in-house and customized software which cannot be transferred to any other controller. Most code developed for PLC's is application specific and cannot be migrated to future applications. In addition to the PLC, the controller language may not be portable to a similar controller. These issues pose a formidable argument for finding a single portable robust language for the entire application.

The diversity of applications within Boeing does not allow for standardizing on a single language or controller. However, a controller with a robust language function base allows for immediate application of skills used with other computer programming languages such as Basic, C, Fortran, and Pascal. Computing iteratives such as FOR, IF--THEN, WHILE, DO and CASE provide high level syntax necessary for control of complex processes. These factors are sought after because they greatly reduce the maintenance costs by providing a common set of characteristics already understood by computer programmers. Another area of concern involves connection through a network to company business systems and storage facilities. The vast amounts of process data being collected and analyzed by upstream systems is transferred using many different network protocols. To provide this function, a controller or host computer must have memory for file storage and control of one or more protocols for file uploading and downloading. Some applications require data transfer using custom protocols developed with the controller language. Many of the older control systems support the crudest of data input and output. This can slow the automation process and also affect overall production costs. The number of process and upstream computer systems involved in the automation process continues to grow resulting in increased layers of software. The software development environment for each layer affects the overall time to production. Software development for the machine controller involves several phases.

After a structured design has been developed, the initial coding phase of all machine functions takes place. Following this phase is test and modification of the software with or without the machine in the loop. At this phase, all subsystem software is individually tested. Integration phase involves debugging all subsystems together with the machine controller. Once the subsystems are connected, all languages must be able to communicate through the main controller. The debugging environment on the controller now becomes a critical tool in testing the system operation.

Multiple modifications to the application software are made by the system programmer during this phase. Continual updating of the application software can be very time consuming depending upon the efficiency of the debugging and programming environments. For example, a compiler based language may be more powerful in terms of functional capability yet continuous compilation, linking and perhaps downloading can be extremely time consuming. On the other hand, an interpretive language can be immediately modified and tested without compilation, or linking. At this point, the use of one language for all subsystems can significantly reduce the programming complexity as well as the manpower required to get the application on-line.

An area often overlooked during this phase of software development is the end user or factory operator. While the efficiency of the development environment plays a significant role in bringing the process on-line, it must also provide a rich graphical user interface (GUI.) Most aerospace automation applications require one or more operators in the loop to monitor the process. The simplicity with which the process can be graphically represented to the operator insures better participation during part manufacturing. An efficient debugging environment for graphic objects such as icons which activate process functions is not available on many control systems.

Once these development phases are complete and the application is on-line, the software maintenance phase is activated. Inevitably, the process requirements change as the product is improved. Modification forces changes in the application software and usually reprogramming of some of the process programs. Here again, the development environment is critical to making rapid changes in the process. A system which supports off-line development and test can be extremely cost effective in the factory environment. Conversely, stopping production to modify and test application code can be costly.

<u>Control System Requirements</u>

The issues of process complexity, control system and language, life cycle costs, and previously successful projects are considered during the planning and design of an automation application. Because of the complexity of aerospace manufacturing, the control system is usually the host in orchestrating a process. There are many simple operations being performed at Boeing requiring PLC's and/or rudimentary control systems. The wide range of complexities of applications forces Boeing to choose different controllers for different applications. Alternatively, standardization of control systems would reduce the level of automation manufacturing by limiting applications to the technological capabilities of the control system.

Advanced applications may require a system which controls 1 or more multi-axis robots and several dependent/independent axes of motion. Dynamic coupling of axes in some applications may also be a requirement. Simultaneous control of serial communications and digital I/O information may be essential. Advanced applications may use machine vision for inspection or vision guided motion. Moreover, a prioritized response to critical interrupts during process execution is usually mandatory. These pre-requisites place a formidable load on any controller.

Factors such as multi-tasking capability, task prioritization, and time slice assignment become fundamental criteria for the controller's operating system. Without these capabilities, the control system cannot effectively perform complex automation tasks. In addition to operating system performance is the efficiency and reliability of internal coupling between hardware and software in a machine controller. The operating system running underneath the language is usually hard coded to motion control boards, digital I/O interfaces, hard disks, and emergency stop circuitry. Multiple microprocessor systems controlling trajectory generation, digital closed loop servo control, external communications, graphics, vision, and power management are all interdependent.

The application complexity determines which of these factors are required to implement the process. Another consideration in controller selection is the number of axes and type of motion required. An application may not have any motion control or it may be a multi-axis machine with vision guided motion. This implies two controllers with very different sets of functional criteria. Thus another key factor in controller selection is the configurability of the system. A control system which can accept a number of optional subsystems to meet different requirements provides a cost effective application solution.

Collecting the topics and issues discussed in this paper provides a general outline of problems which exist in the manufacturing environment. There are still more problems and new solutions being developed today in factories around the world. This paper is not intended to be a catch all of automation issues, but an insight into the growing complexity of factory automation. The next section discusses four case studies of systems currently in use within the Boeing company. The general system block diagrams are presented with a discussion of some problems and solutions related to each system. The exact details of the application are omitted in order to protect any proprietary information.

Case Studies

System 1

System 1 uses an Allen Bradley 9/260 series controller to perform processes on stringers and stringer clips. The system executes RS274D coded programs and controls two axes of motion using incremental encoder feedback for positioning. Figure 1 depicts the hardware block diagram for this system.

The operator control panel is part of the control system. This controller has two serial communications port,: one for DNC downloading of part programs from a file server, the other retrieves data from a thermal scanner. A specific DNC protocol had to be adhered to in order to

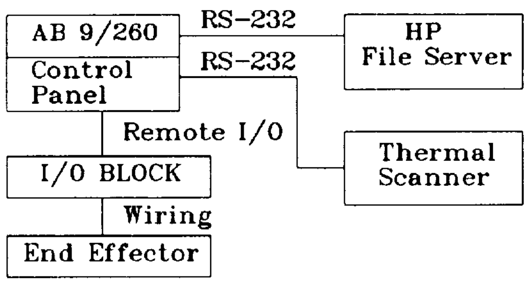


Figure 1

Number of axes	2
Number of I/O's	60-70
Number of serial ports	2
End Effector	1
Languages	PAL, SIPROM
Lines of code	800-1000

Table 1

transfer program files. A network analyzer was used to re document and debug the transfer protocol. Only one RS-232 port can be used at a time, as the second port is not a fully functional RS-232 port. The communications protocol is specific to AB. Different ASCII characters sent to this port cause predefined functions to occur. Thus, the limitations of the communications set reduced the overall flexibility of the system while increasing development time.

The application language for this system is ladder logic (PAL). The development environment consisted of separate software packages provided by Allen Bradley. PAL code was developed off-line on a PC using an AB editor package. The software was then downloaded to the AB 9/260. Debugging was accomplished by running the PAL programs while monitoring the process on a remote PC. The application code could not be single stepped for debugging. The monitor process can be started and stopped only. Motion parameters include: gains for P, I, & D, gain break-point parameter, following error limit. There are no pole or zero adjustments for the digital closed loop servo control.

System 2
System 2 uses conventional cutters mounted in an electric router to trim the periphery of composite parts for aircraft. Figure 2 depicts the hardware block diagram for this system. The part periphery are defined to tooling edges where a robot slides a router bushing. This system uses a CimCorp CimRoc4000 controller to perform all robot motions. In addition, the router motors have controllers to perform all router sensing and control of the electric routers. Material handling shuttle tables are controlled by PLC's based on digital signals from the robot controller. There are over 128 digital input and output points defined and three serial ports for the printer, router controller and position probe. Software was developed in C, running under DOS.

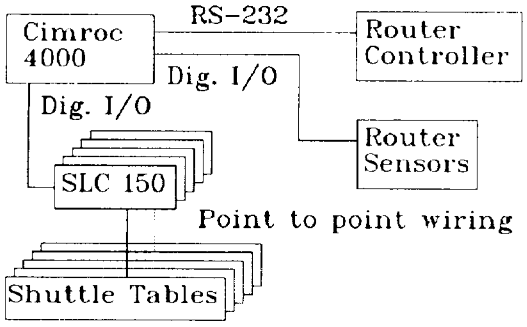


Figure 2

Number of axes	7
Number of I/O's	200
Number of serial ports	1
End Effector	Multiple
Languages	C
Lines of code	30,000+

Table 2

The DOS/C development environment made use of existing skills to efficiently implement a number of operator security functions. Graphical user interfaces were developed with the aid of a commercial graphics package and libraries for serial communications and ISAM databases were used extensively.

The most severe limitations were associated with the use of a single tasking operating system (DOS). Minor difficulties were encountered with network communications owing to interrupt collisions between the network card and communication cards needed to direct the motion control cards in the real-time back plane.

System 3
System 3 utilizes an AB 8600 controller interfaced to a 7-axis JOBS Jomach 16. This controller manipulates the Jomach 16 as well as various end effectors used in fuselage assembly processes. Figure 3 depicts the hardware block diagram for this system.This application also uses 3 PLC's, one for interfacing to a tool storage/retrieval rack, and two others for controlling the position of tooling headers. All three PLC's are connected to a host AB8600 using "Data Highway". Each of the PLC's uses "Remote I/O" for inter-PLC communication.

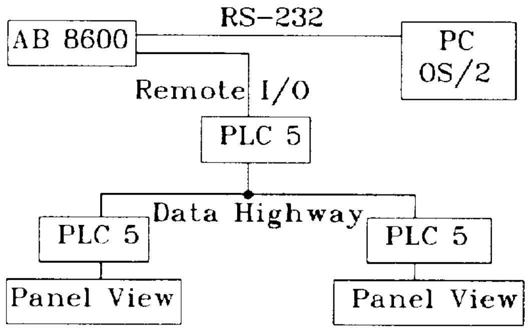


Figure 3

Number of axes	9
Number of I/O's	200
Number of serial ports	1
End Effector	Multiple
Languages	PAL,SIPROM
Lines of code	6,000+

Table 3

This system required dynamic coupling of axes during end effector drop-off and pick-up. The controller provided this capability through hardware partitioning of the axes. Memory on the 8600 CPU was also partitioned and used for up to 5 different tasks. Dynamically coupled motion was achieved using Allen Bradley's Axis Manager software.

The complexity of the application required the use of PLC's in addition to the system digital I/O blocks. Because of the difficulty in programming the PLC interface with the operator console, two Allen Bradley "Panel View" systems were used. The PLC's use "Remote I/O" to communicate with the operator consoles, and discrete I/O to activate motion control cards.

The system integrator used Siprom and ladder logic languages to implement all process functions. The serial communications protocol used by the AB8600 is specific to the controller, and it was necessary to use a network analyzer to determine how to implement reliable communications with the AB8600. Programming tools for graphics display were inflexible and poorly documented. All GUI's and interfaces with the 8600 CPU card cage were controlled via a PC.

Since the response times for probe contact were inconsistent, programming custom probe routines for probing normal to a surface was particularly difficult. Machine motion in some applications was not as smooth as expected, due to the length of the SIPROM code and the loop execution time.

To interface a thermal scanner in this system required a usable serial port. Further, coding of customized M-codes routines in SIPROM were required for retrieving and computing the thermal data.

File operations have some minor restrictions. Downloading of files is limited to 6 ASCII characters for file names. Formatted file lengths are limited to 255 records (132 characters per record). This forces new data files to be created each time the 255 record boundary is filled. Also, any formatted file read by the 8600 CPU cannot be larger than 255 records. The record size constraint creates further overhead in uploading data files from the AB 8600 to company business systems. NC part program files are unformatted so they can be as large as memory allows. Deletion of files requires a manual key insertion and editing privileges. Thus, operator lockout was not possible, so data integrity could not be assured. ie; operator can modify production files.

Software maintenance is difficult and costly due to the structure of SIPROM code and the size of the PAL code running on the PLC's. The single biggest problem with this system is lack of memory. The machine controller is running at maximum capacity. Because additional memory is unavailable, no new process can be added to this system. For example, adding another RS-232 port would require memory to set up a serial communications structure. Any modifications to existing code is very difficult. On the other hand, this system is currently exceeding production goals in the factory.

System 4

The retrofit system consists of a 5 axis JOBS Jomach 16 with a sixth W feed axis, and a spindle. This system is interfaced to an Adept A-series IC controller. The purpose of the retrofit system is to provide a test and feasibility workcell for various automation processes under development within Boeing. Figure 4 depicts the hardware block diagram for this system. The system goal was to be extremely flexible, accommodating diverse applications.

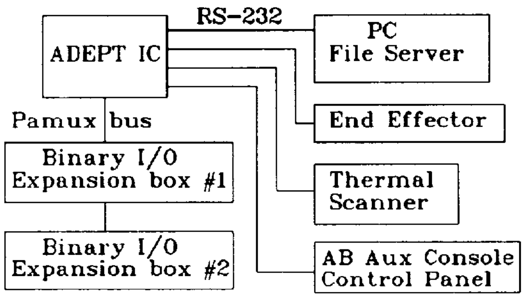

ADEPT IC
RS-232
PC File Server
End Effector
Thermal Scanner
AB Aux Console Control Panel
Pamux bus
Binary I/O Expansion box #1
Binary I/O Expansion box #2

Figure 4

Number of axes	7
Number of I/O's	100+
Number of serial ports	1
End Effector	Multiple
Languages	V/V+
Lines of code	15,000+

Table 4

The system uses 4 serial communications ports. One is connected to an external PC for file transfer. Another is connected to an operator control console (OCC). The third is connected to a thermal scanner, and the last is used for communicating to an end effector control system.

The system uses more than 100 digital I/O's for process control. Most of these are used in control of spindle operations. Digital I/O is split into three groups: input, output, and interrupt functions. Each of these groups can be subgrouped into banks of 8 discrete I/O's for partitioning in software. The interface to the OCC uses both RS-232, interrupt, and digital inputs. Because cycle start and cycle stop functions are critical to NC operations, a non-maskable interrupt is used to acknowledge input form the OCC.

The Adept controller provided many functions used in serial communications. For example, file transfer functions from the PC to the Adept are buffered. Although an in house transfer protocol is used, Kermit or Xmodem could have been applied. Because the amount of data read from the thermal scanner is small compared to file transfer, communication is done asynchronously without buffering.

The retrofit project benefits from using one language capable of controlling I/O's, interfacing to an operator console, defining serial communication formats, and developing decision paths for the application software. The language is efficient in supporting variable definition. For example, a program must perform automatic range changing of the spindle drive gearbox. The application code was written using variables such as **sp.in.rng.1**, and **sp.in.gear1.i** to define the spindle gear range and state of the gear 1 input sensor.

The tools for graphics were used extensively in developing user interface screens. Features such as buttons, icons, window and scrolling were implemented in most of the application software. The language also supported structured techniques which allowed for modularizing the application code. Because of this, many code modules are being reused in other applications currently under development. On the other hand, the language V+ is proprietary to the controller and required some training before programming could begin. The controller fully supported RS-232 and file transfer functionality but was not equipped with protocols such as Ethernet, SNA, or MAP. This shortcoming provided difficulty in interfacing to company business systems.

Maintenance and life cycle costs of the software are difficult to determine because code is always being developed for new applications. It should be noted that by developing modular functions and meaningful variable definitions, most of the application code is understood by reading it directly. Electrical maintenance of the system is undetermined because the machine has not broken down yet. Mechanical functions remained the same after the integration.

NC Translator Application

In addition to the four previous case studies, there was a requirement to develop an NC translator which could read NC code developed for system 3 and execute it on system 4. The application required exact replication of NC motion with a control system using kinematic trajectory generation. The Adept controller uses built-in kinematics during trajectory calculations. The kinematic definition of the machine includes link lengths, joint angles, joint configurations etc. The NC translator application required encoding the NC joint positions into WORLD coordinates for use by the control system's trajectory generator. An NC controller moves the machine joints to locations using linear or circular interpolation. The G-codes being executed by the NC machine determines the type of interpolation employed. Conversely, a robotic controller uses kinematics to compute trajectory points for driving the tool tip. The robotic controller can then use linear or joint interpolation to drive the machine in WORLD, TOOL or JOINT space.

Path motion created unique problems with respect to accuracy. A path may be represented by a series of consecutive points. As the tool tip moves through these points several events occur. The tool tip moves toward point 1 while the control system is computing a trajectory for point 2. As the tool tip approaches the target point 1, it may move through that point or come close to it as it moves towards point 2 in the path. The controller looks ahead 1 point in the path and computes a trajectory to that point. At some time in the trajectory, the tool tip begins to move towards point 3 and so on. The velocity and acceleration values directly affect the accuracy of the tool tip in following the prescribed path. In machine routing, the smoothness of the motion over a path is critical to the quality of the new surface left behind by the router blade. A constant velocity is required to make a smooth cut.

The controller allows for tuning envelopes around endpoints in motion but did not allow for definition of a tolerance envelope around path points. A solution required close spacing of path points in the NC program. During path motion execution, the next point in the path was broken down into a series of smaller constant velocity moves. The machine structure of 5 axes together with path slicing computations produced two wrist configurations for the same point. Additional software was written to assure wrist configuration was maintained during path motion. The result allowed the machine to follow paths dictated by RS274D G-codes even though the trajectories were computed using forward and inverse kinematics.

The NC translator requirements included simultaneous execution of the following functions: a graphics display including which NC block was currently executing; real time monitoring of an auxiliary operator control console; preparation of path points for tool trajectory; executing proper motion as defined by RS274D G-code standards.

The application required the use of 3 tasks and several internal software flags for inter task communication.

The multitasking capability of the operating system was invaluable in coordinating the 3 application tasks. Software was used to set task prioritization and optimize stack sizes. This application is currently used to test NC programs for developmental assembly concepts.

Future Work

If robots and machine tools are to realize their full potential, controllers must improve their computational performance, support reusable software and provide for system extensibility. Open architecture controllers based on accepted industry standard hardware, operating systems, and application languages are arguably the best way to support these improvements.

Machine controllers are typically two generations behind the best available microprocessors. This performance lag occurs due to lack of portability of control software as well as robot and machine tool suppliers using proprietary high level and assembly programs to implement unique mini-kernels in lieu of a conventional operating system. Control software written in ANSI C with careful conformance to POSIX standard system calls can be ported to new processors in a matter of days. The use of standards further encourages software re-use, since application code can often be re-compiled in the new environment and linked into higher level software designs.

Robot system extensibility demands a computing hardware environment that enjoys high volume use and a spirited development community to ensure an uninterrupted stream of hardware to support emerging requirements.

Boeing, in support of this approach, is developing open architecture controllers and motion control libraries in cooperation with several commercial vendors. The robot controllers are VME based, programmed in ANSI C and are POSIX compatible. Extensions to this work will provide retrofit software applications to ease the adaptation of open controls to new machines. Servo tuning tools, simulation systems, calibration applications, and upstream system interface libraries will be developed during the next year or two.

The author would like to thank the following dedicated automation and robotics engineers at Boeing for their experienced input: Craig Battles, Rich Morihara, Stan Munk, and Scott Muske.

References

[1] Reintjes, J. Frances. 1991 *Numerical Control: Making a new Technology*, New York. Oxford University Press.

C-2.

6031

AIAA-94-1186-CP
N94-30536

INTEGRATION OF VISION AND ROBOTIC WORKCELL

T. A. Bossieux
Ford Motor Company
Alpha Manufacturing Technologies Applications Center
Dearborn, Michigan

P. 5

Abstract

The paper discusses the incorporation of vision into a robotic cell to obtain cell status information and use this information to influence the robot operation. It discusses both mechanical and informational solutions to the operational issues which are present.

The cell uses a machine vision system to determine information about part presence in the shipping tray, part location in the tray, and tray orientation. The vision system's edge detector algorithm is used to identify the orientation of the packing trays. In addition, different vision tools are used to determine if parts are present in the trays based on the unique configuration of the individual parts.

The mechanical solutions discuss the handling of medium weight (10 - 25 lb.) parts at an average cycle time of 3.1 seconds per part. The robot gripper must handle 33 different models, three identical parts at a time. This is accomplished by using stacks of rotary actuators and slides between the stacks.

I. Background

One of our manufacturing divisions was having an ergonomics issue with their alternator packing operation. The pack operator was required to manually handle 500 15-pound parts per hour. In addition, he was required to handle one 25-pound, 22 inch by 44 inch shipping tray for every 15 parts.

They requested assistance with the development of a robotic cell to unload their final test line, place the parts into shipping trays and handle the shipping trays. A dunnage transporting conveyor was already present, however, it was manually controlled.

There are only two different rating sizes for the alternators (95 amp - Medium Frame (MF); 130 amp - Large Frame (LF)). However, there are 33 different types of alternators with the differences being mainly in the mounting configurations. There are 18 different possible combinations of orientation moves from the test line to the shipping trays (3 positions on the test line, 3 positions on the holding fixture, 2 different tray orientations).

The alternator is assembled using three through bolts (see Fig. 1). These bolts define the three points at the bottom of the alternator. The shipping trays have the three points contained and supported for shipping.

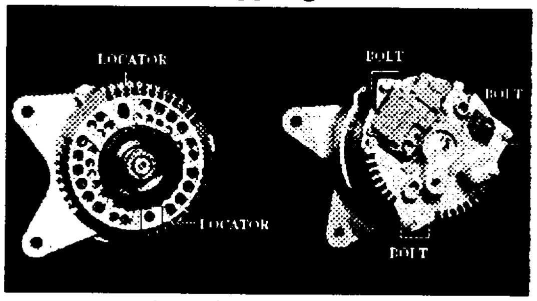


Fig. 1 Alternator

The original layout of the packing cell is shown in figure #2.

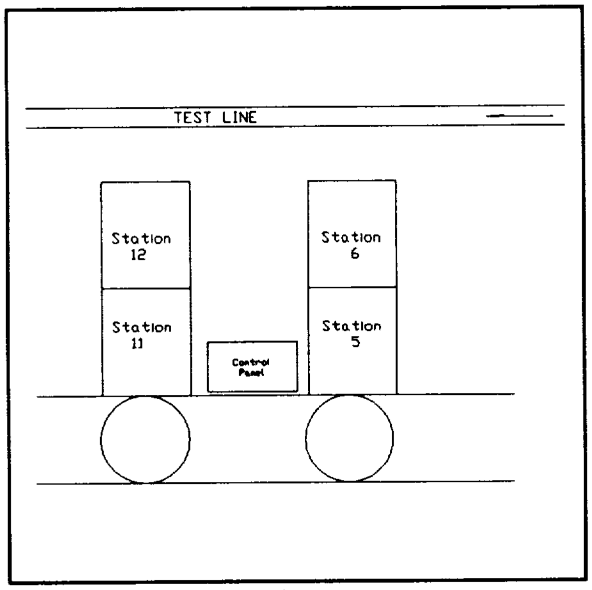


Fig. 2 Layout

II. Mechanical Issues

A. The following is a partial list of some of the major mechanical problems affecting the automation of this operation:

1. The shipping trays cannot be modified in any way. These trays were not designed for automation;
2. There are three different orientations (to the part locators - see Fig. 1) of the parts into the trays;
3. The part spacing in the tray is different for the medium size alternator (medium frame) and the large size alternator (large frame) by .030 inch (0.76mm);
4. Some of the parts have interference fits into the trays;
5. Because of the non-synchronous operation of the test line, three different parts could be waiting at the unload station at a given time.

B. Solutions to the above issues are described below :

1. Because of the number and cost of shipping trays in use, they can not be modified in any way. There are five different types of trays based on the different sizes and mounting configuration of the parts. The trays were not designed for automation and the standard grip points are 44 inches apart.

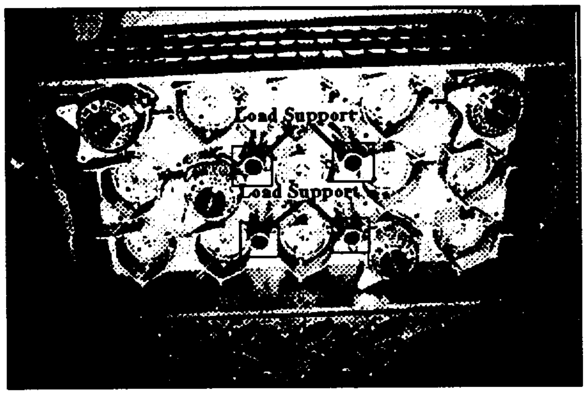


Fig. 3 Shipping Tray

As the 44 inch spread between the tray grip points would make the robotic gripper very large and heavy, another grip location was a necessity. The only common internal features to the different trays are the four load support posts which are hollow (Fig. 3).

2. Friction gripper devices utilizing urethane die strippers and individual remote center compliance devices (Fig. 4) were developed to work inside the hollow posts (Fig. 3). This friction gripper demonstrated capability of moving over 100 pounds while maintaining enough stability to directly place the tray into its next position.

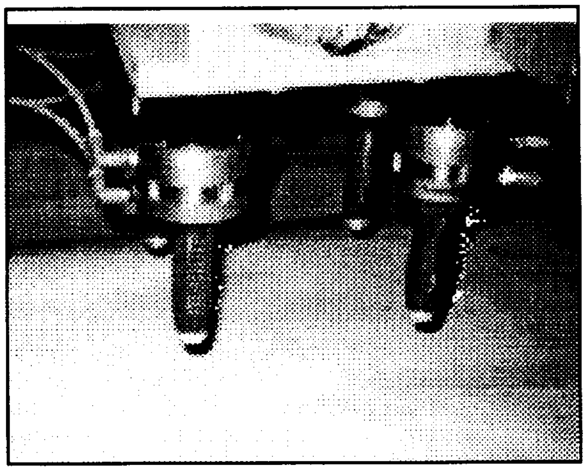


Fig. 4 Friction Gripper

3. As the parts have different mounting configurations, there are

three different possible orientations of the parts into the shipping trays. The part orientation (to the machining locator holes in the top - Fig. 1) in the test line pallets is not consistent between different part types. The part orientation in the shipping tray is also not consistent between the different part types.

The multiple orientations required would be a simple task for a robot handling a single part. However, the speed required to meet cycle time, in conjunction with handling the trays prevents this from being a single robot system. The initial solution was to have the parts removed from the test line, orientated into tray orientation and placed into a 3-part holding fixture by a robot. Then a second robot would pick up three alternators at a time and place them into the shipping trays. It would also handle the empty trays.

After a ROBCAD ™ simulation showed that this process would only achieve a cycle time of 5.0 seconds, an additional small robot was added to the system. There are now 2 small robots removing parts from the test line, orienting them and placing them into four (4) 3-part holding fixtures. (Fig. 5)

The final layout of the packing cell is shown in figure 5. The robots, vision system and escape line were added to automate the cell.

Based on this process, the third robot's gripper must handle three parts simultaneously. The gripper must also be able to handle the three different orientations for part placement into the trays. Changing orientation was accomplished by using a stack of two Robohand Ultra Thin Rotary Actuators (RR-46) capable of 180 degree rotation in each of the three individual part gripper stacks (Fig. 6).

4. The spacing between the LF parts in the trays is different than

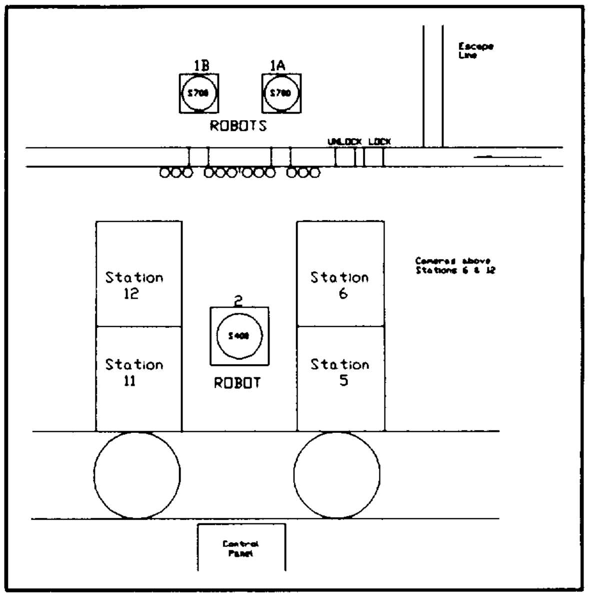


Fig. 5 Final Layout

the spacing of the MF parts by 0.030 inch (0.76 mm).

The three-part gripper was designed to change the distances between the individual grippers by using two THK slides with actuator cylinders on each side (Fig. 6). Identification of part type is discussed in Informational Issues.

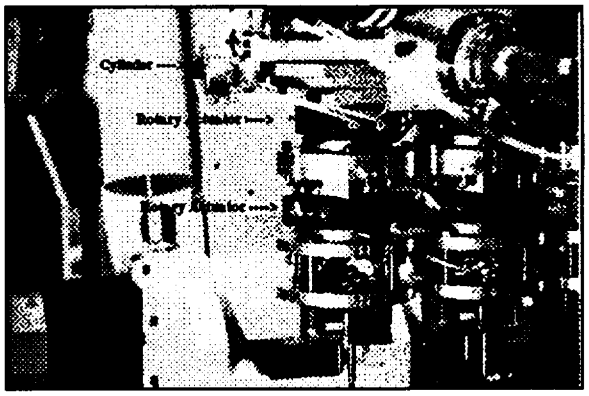


Fig. 6 Gripper

5. Some of the parts have interference fits into the trays. Because of this, individual stack compliance devices were added. A simple machined cone and spring tension were used for the required compliance (Fig. 7).

6. Because of the non-synchronous operation of the test

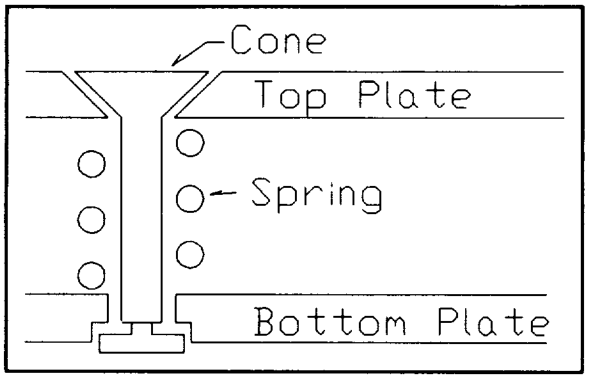


Fig. 7 Compliance Device

line, three different parts could be waiting at the unload station at any given time.

An escape line will be added to the test line to hold the third part type when needed. The other two parts will be handled by loading one into trays at Station 6 and the other at Station 12 (Fig. 8). This is the maximum number of different parts which would be at the unload station under normal operating conditions.

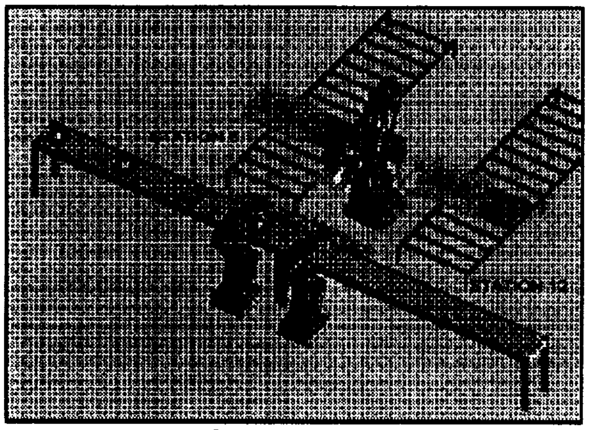


Fig. 8 ROBCAD DRAWING

<u>III. Informational Issues</u>

A. The following is a partial list of some of the major informational problems affecting the automation of this operation:

1. There are no features on the trays to insure they come to the pack station in the same orientation;
2. Because of changeovers and system fallout, a tray may return to the pack cell partially full;
3. Part type identification is required as the parts enter the unload cell to insure proper handling;
4. Each tray must be confirmed as full prior to leaving the cell.

B. Solutions to the above issues are described below :

1. There are no orientation features on the tray to insure that they are stacked in a consistent orientation. Therefore, they can arrive at the cell rotated 180 degrees to each other. As the parts can only be correctly placed into the trays in one direction, the robot must know the orientation of the tray.

Vision has been used for inspection and location determination for many years. In this application, vision is used mainly to gather cell status information.

As in most production situations where vision is used, lighting is critical. The selected system uses an intensity meter and stops operation if the lighting falls outside acceptable ranges.

In order to determine the orientation of a tray the vision system's edge detector is used in two opposite locations. This tray has webbing which is missing at one of the corners. Tray orientation is determined based on where the webbing is found and, as a safety for broken trays, where it is not found (Fig. 9).

2. Because of changeovers and system fall out, a tray may return to the loading cell partially full. The system must be able to identify the position of parts in the tray to prevent the refilling of those positions.

In order to identify where parts are in the trays, the vision edge detector was tried first. Because of the large number of air holes in the

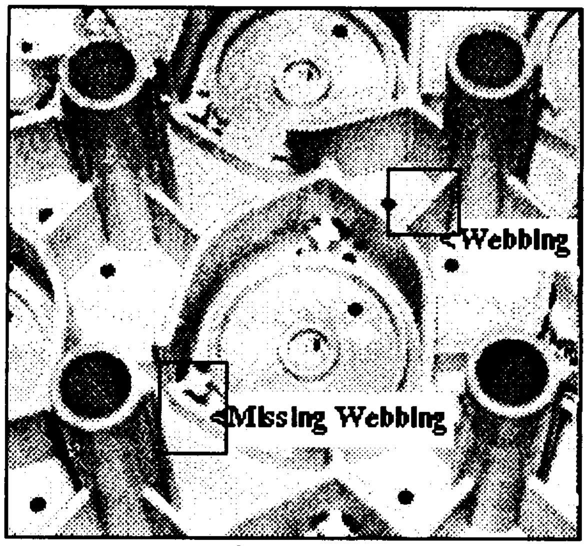


Fig. 9 Tray Webbing

top of the alternator, a find/not find limit was very robust. However, because of a concern for debris in the tray, this method is not usable (a crumpled 8.5" x 11" piece of paper had approximately the same number of edges). The system was changed to identify a specific feature, such as the diameter of the alternator pulley, to insure correct identification of part present.

In order to communicate with the robot, a method for identifying the specific location in the tray was developed and is shown below (Fig. 10).

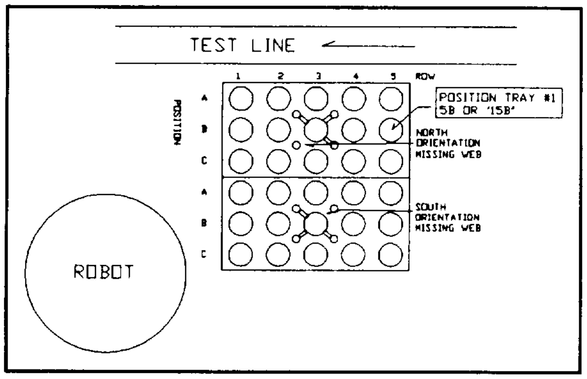


Fig.10 Part Location Identification

The large robot needs to know where parts are present in the next tray row prior to removing parts from the holding fixture. This was handled by using digital I/O with one vision system output for each of the three positions in a row and by having the robot request information for the next row immediately after releasing parts in the previous row. The robot used five outputs to request information from rows 1 to 5.

3. The first robot must know what part is presented by the test line so the robot can properly orient it for insertion into the holding fixtures, or to diverted it to the escape line, or to a specific robot if the cell is unloading more than one part. This is accomplished by using the test line pallet magnetic information card and a reader at the unload cell.

4. The customer requested that each tray be confirmed as full prior to being released for shipping.

At the completion of loading row five of a tray, the large robot will request that the vision system reconfirm that all 3 tray positions contain a part for all five rows. If a position is missing a part then the system will stop operation and notify the tender.

IV. Conclusions

The final system will use three robots and one 4-camera machine vision system to handle 15,000 parts per day.

The use of a multi-purpose gripper to handle both multiple parts and the shipping trays will allow the cell to achieve a average cycle time of 3.1 seconds.

This process development shows the benefit of using machine vision to solve cell informational issues. The use of machine vision easily solved complex informational issues which would have required many elaborate and costly sensors to accomplish.

Note: ROBCAD is a Trademark of Technomatix Technologies, Inc., Novi, Michigan.

S1-63
207634
p- 4

6032

AIAA-94-1187-CP

N94- 30537

Meeting the Challenges of Installing a Mobile Robotic System

Celeste DeCorte
Cyberomotion, Incorporated
Salem, Virginia

Abstract

The challenges of integrating a mobile robotic system into an application environment are many. Most problems inherent to installing the mobile robotic system fall into one of three categories:

- **The physical environment** — location(s) where, and conditions under which, the mobile robotic system will work
- **The technological environment** — external equipment with which the mobile robotic system will interact
- **The human environment** — personnel who will operate and interact with the mobile robotic system

The successful integration of a mobile robotic system into these three types of application environment requires more than a good pair of pliers. The tools for this job include: careful planning, accurate measurement data (as-built drawings), complete technical data of systems to be interfaced, sufficient time and attention of key personnel for training on how to operate and program the robot, on-site access during installation, and a thorough understanding and appreciation — by all concerned — of the mobile robotic system's role in the security mission at the site, as well as the machine's capabilities and limitations.

Patience, luck, and a sense of humor are also useful tools to keep handy during a mobile robotic system installation.

This paper will discuss some specific examples of problems in each of the three categories, and explore approaches to solving these problems. The discussion will draw from the author's experience with on-site installations of mobile robotic systems in various applications.

Most of the information discussed in this paper has come directly from knowledge learned during installations of Cybermotion's SR2 security robots. A large part of the discussion will apply to any vehicle with a drive system, collision avoidance, and navigation sensors, which is, of course, what makes a vehicle autonomous. And it is with these sensors and a drive system that the installer must become familiar in order to foresee potential trouble areas in the physical, technical, and human environment.

Physical Environment:

What you see is not always what your robot sees. Picture a hallway 5 feet wide, carpeted, and 30 feet long. Problem or no problem? Usually it's not as easy as you think. I can walk down this hallway easily, why can't the robot. Turn out the lights or have a few drinks then try to walk down the hallway. Stubbed your toe didn't you. We now have established that even humans can have a problem walking down a hallway, and we have a self-righting mechanism: arms. What are the important aspects of the physical environment to a robot? Anything that can affect navigation or collision avoidance, such as floor surfaces, walls, and obstacles.

Now that we know the potential hazards, let's start from the ground up. Different floor surfaces cause problems that are specific to each robot and are mostly dependent upon the type of drive system. If you could require, by law, that all buildings use only one floor covering, which one would you choose? carpets? wood? tile? or that bumpy pebble-type floor? Any robot installer in her or his right mind would say tile. It's mostly level, no bumps, and has a low coefficient of friction. Tile and poured concrete floors exist in great quantities in the manufacturing world, but not so much in the corporate world. Carpeting has its ups and downs and is very deceiving. Most installers see thick plush carpet and hit the roof; they see indoor-outdoor carpet and get warm fuzzies. Maybe not! Each carpet will have its own unique problems related mostly to the under flooring, which you can't see. Some carpets will give back a sonar image; some will slip on the floor; some will act just like a tiled flat floor. Until you've run over it a few times, you'll never guess what troubles the floor will create. Carpet may affect your navigation through slippage, your collision avoidance through sonar reflection, and your drive system through increased frictions. Other than that, carpet is a wonderful surface for robots. Most of the same problems will recur for one of those pebble aggregate concrete floors. They are very uneven and generate a great vibrational analysis atmosphere for your system. This floor could be very hazardous to your electronics. No floor is ever simple. Even poured concrete may have slopes, cracks, and dips. Although working on these surfaces may not be simple, remember that all these floor types have been -- and are being -- traversed by autonomous vehicles.

After you've mastered the floor in your building, it's time for the walls. I'm sure you are wondering how walls could cause a problem. It's the type of material they are made of, as well as what people put on them, that create your problems. Sound-absorbent cubicles, sheet rock, and concrete make up most walls in buildings.

Sound-absorbent cubicles may create a problem depending upon the wave-length of your sonar. These walls do not reflect all wavelengths of sound, and that is a problem. At the frequencies used by Cybermotion this is seldom a problem. Another problem with these cubicles is that they are easy to move and may never be in the same location from one day to the next; therefore, they are not good navigational walls. If you have no other options they are better than not navigating at all.

Sheet rock is wonderful, usually the best surface that you can imagine. The only down side is that everyone loves to mount items on this type of wall. Door moldings, fire hoses, fire extinguishers, water fountains, and many other objects. The resulting corner reflectors, as I like to call them, give an excellent sonar echo return that makes mountains out of mole hills. In a wide hallway where your vehicle has room to pass these objects will not be a problem. But in tight hallways you may choose to avoid these areas rather than reduce your safety by reducing collision ranges.

The third type of wall is cinderblock. These concrete building blocks are full of holes and bumps. This rough surface generates some interesting echoes and their effect definitely relates to how you use sonar in your system. The Cybermotion SR2 can be modified to ignore the false images that are returned from such a rough surface. If you don't navigate using walls then this part doesn't really matter.

So far we've discussed what's below, (floor) and what's to the side, (wall). All that's left is what's in front of you, (objects). Walls and floors affected our drive system, our collision avoidance, and our navigation sensors. Obstacles affect our collision avoidance sensors. There are two different types of obstacles: fixed and floating. Every building has it's unique fodder or floating obstacles. These include: mops, displays, decorations, etc . During your walk through of the facility you will see where these obstacles will generally be located and you can plan accordingly. If you can find out what day is trash day I recommend that you visit the day before to see everything at it's worst. People are creatures of habit, and once you learn their habits, you can plan around them. In one particular instance a hallway was full of furniture. I thought it would be moved into someone's office, but six months later I have been assured that it is still in the hallway and is not going anywhere in the near future. Fixed obstacles are no problem, but make sure that the vehicle has sufficient clearance to move around them.

Once you figure out where you want your vehicle to travel based upon your information about floors, walls, and obstacles, the last piece to the physical environment you need is an accurate map of the facility showing fixed obstacles and hallways. You may be surprised to know that "as-built" drawings rarely exist; they are more like "as-planned." You may need to do some measuring to get the maps up to date. Programming and debugging are much simpler with an accurate map.

Once we've mastered the building or physical environment, our system is operating on single floors, our paths are debugged, and the vehicle is working perfectly, it's time to tie everything together through the technical environment.

Technical Environment:

The technical environment is made up of external equipment with which the mobile robotic system will interact. These can include: doors, elevators, lighting systems, etc. The question here is: what must I control to run all of my paths with one robot? At Cybermotion we don't normally offer robots equipped with door openers and button pushers as an option. So far these options are extremely cost prohibitive and power hungry. Other companies may have this option available. So the trick is to call the elevator to the right floors and get the door at the end off the hallway to open and close at will. Many different approaches have been taken to solve these problems. There is the "We'll make sure that the door is propped open when the vehicle is running" approach. These good intentions can work if it is someone's specific job to make sure that the appropriate doors are open and that you don't violate any fire codes. Otherwise it will not get done all the time, and you have to develop a solution. You could choose to install motion detectors to automatically open and close for any movement. This option requires minimal installation but is a potential security problem if you want to restrict access to the area. Your options are limited by your customer requirements. One option is an automatic door with IR or RF receiver and a transmitter on the vehicle to signal when the robot requires the door to be open and closed. This option requires integration of inexpensive hardware on your vehicle. If hardware integration is not desirable, door access can be controlled at the base station computer. This option requires a communications link between the door mechanism and the computer. Part of the program sent to the vehicle would be to open the door at a certain time in the program. Each option is viable; it's just a matter of deciding which one best matches the job.

Personnel intervention, RF or IR link, and base computer control, the options discussed above, can be applied to most technical environments. Even elevators can be handled in this manner. Personnel intervention should be used only if there is already an elevator operator in the building. Working with elevators will require a controls interface provided by the elevator manufacturer. Another option is to install a poking device with vision recognition to ensure that the vehicle gets off on the correct floor. This poking mechanism will greatly increase the cost of the robot. Both the IR or RF and the computer base station will require information to be received and transmitted to the elevator controls. You will need an architecture that will operate like your button-pushing finger. Consider all the mechanisms with which the system will be required to interact, and pick a solution that can best handle all interfaces. Some day the button-pushing finger may best suit the job, but that day has yet to come. There is one more interface that requires a special interface. The people interaction.

Human Environment:

The Human environment is made up of everyone that could possibly come in contact with the robot. As you might have guessed, most of the biggest problems you will have to overcome are in this environment. There are three basic groups of people that you will need to work with: those who do not interact, those who modify the environment, and those who operate the system. Every step of the way, you will have people to train and you will have to explain the operation of your system to everyone from the janitor to the president of the company. Each system comes with a certain amount of training. Usually the more you know the more effective your system. What you don't know can hurt you just as much as what you do know.

People who come into your facility while the robot is running; such as visitors, employees that are working late, and contractors, are those who do not interact with your system. These people typically exhibit a facial expression of amazement followed by the long stare. This curiosity response as I like to call it only lasts for about one minute. After such time they consider the vehicle part of there environment and ignore it like everything else.

Those who modify the environment, such as cleaning people, can inadvertently create a difficult operating environment for vehicles. Buckets, mops, trash cans, boxes, and vacuum cleaners are among the obstacles that you may have to avoid or maneuver around. The best solution allows everyone to complete his or her job with minimal changes. Habits are hard to break. If Mr. Clean always leaves his tub-o-trash in front of the elevator, this is a problem. Give him a little information about how the vehicle operates, and possibly a note from his supervisor, you can start to work replacing habits that may inhibit vehicle operation with habits that facilitate operation. Once the habit is changed, it's all down hill.

The operator may be the easiest -- or the hardest -- individual to get to cooperate. Some people love technology and will be hanging on every word about the system. Some are absolutely frightened. Some just believe that robots are replacing people; such people can make your life miserable. Those who love technology are very helpful and usually fast learners, although their curiosity usually generates the need for a few solutions. "I wonder if it can roll over ______" or " What happens if I push ______". Curiosity can kill a robot. Those who have a slight fear of technology can become your best operator. Patience is required up front, but once they see that the robot does not fall apart when they touch it, they get the bug to learn. Best of all they become great teachers to those who come on board after you leave. Then you have the potential spoilers. You cannot force technology on people. Time may bring these people around to your way of thinking, but the best you can hope for is that they don't want to sabotage your project.

Robot installation is a test of skill, knowledge, finagling, and endurance. When an installation is up and running, and you're no longer needed for a helping hand, it is a wonderful feeling that I hope you will get to experience.

S/2-63
30635
p. 6

AIAA-94-1188-CP

N94-30538

DESIGN OF AN AUTONOMOUS EXTERIOR SECURITY ROBOT

Scott D. Myers
Robotic Systems Technology
Westminster, Maryland

Abstract

This paper discusses the requirements and preliminary design of robotic vehicle designed for performing autonomous exterior perimeter security patrols around warehouse areas, ammunition supply depots, and industrial parks for the U.S. Department of Defense. The preliminary design allows for the operation of up to eight vehicles in a six kilometer by six kilometer zone with autonomous navigation and obstacle avoidance. In addition to detection of crawling intruders at 100 meters, the system must perform real-time inventory checking and database comparisons using a microwave tags system.

I. Introduction

High dollar and sensitive assets stored within U.S. Government warehouses, ammunition bunkers and storage yards are vulnerable to a skilled intruder attempting to steal, sabotage, embarrass, terrorize or exploit the U.S. Government during peacetime. Targets range from classified documents, electronic equipment, personnel and small arms to nuclear and chemical material.

General Accounting Office (GAO) report NSIAD-92-60 notes that the Department of defense (DoD) is losing millions of dollars of inventory per year and conducts physical inventory audits that vary by several billion dollars from year to year. This problem is being exasperated by the reduction of security and inventory personnel due to the downsizing of the DoD budget. A highly secure autonomous intrusion detection systems (IDS) using robotic technology would protect these assets in addition to performing a physical audit of inventory on a daily basis.

This system called the Mobile Detection, Assessment and Response System, or MDARS, consists of two parts - an autonomous interior security robot and an autonomous exterior robot. In October of 1993 Robotic Systems Technology (RST) was awarded a three year contract by the Program Manager for Physical Security Equipment, located at Ft. Belvoir, VA, to develop and demonstrate an autonomous exterior security robotic system called MDARS-E.

II. Operational Environment and Concept

The MDARS-R system will be required to operate within fixed areas such as storage yards, office parks, dock facilities and air fields, both within and outside the Continental United States. Majority use will be in areas that are semi-structured with clearly defined boundaries. Within these areas will be structures of many shapes and sizes. The system will be required to operate on concrete and blacktop roads, crushed stone roads, or semi-flat rough terrain, and have the ability to cross railroad track or other small obstacles. Most of the areas will be limited access areas, with only security vehicles allowed after duty hours. Operations will be 24 hours a day in fog, rain and snow conditions.

Up to eight MDARS-E will be operating simultaneously in a six kilometer by six kilometer area or zone. This area will consists of a mixture of different storage bunkers and facilities and warehouse areas. Following a random path, a system will be autonomously looking for intruders or performing barrier and/or inventory assessment on the storage facilities.

During this time, video and status data will be continuously relayed back to the control station for potential collection, however the operator will not be actively involved with any of the systems. His job is to respond only if an anomaly is detected. Once an anomaly is detected by the system, the operator is alerted. He will then take over control of the system via teleoperation for final assessment. If he decides that a false alarm situation occurred, he will put the system back into automatic mode. However, if a real problem is detected, the operator can use the MDARS-E vehicle to respond to the threat or he can send in a manned patrol unit.

III. Requirements

The MDARS-E Requirement document and

the draft Concept of Operations paper define the following requirements for the MDARS exterior system :

- Simultaneous autonomous operation of up to eight MDARS-E systems within a six kilometer by six kilometer zone.
- Be able to travel both random and deterministic paths on road and rough terrain.
- Have a navigation system accurate to less than 0.3 meters within the six by six kilometer zone.
- Normal operating detection speed of 5 kilometers per hour (1.4 meters per second). Maximum teleoperation response speed of 25 kilometers per hour.
- Detection of crawling and/or running intruder from 2 to 100 meters over a 360 degree horizontal field of view.
- Probability of intruder detection between 90 to 95 % with no more than one nuisance/false alarm per platform per eight hour shift.
- Have an intruder detection system capable of penetrating smoke, fog, dust and precipitation.
- Provide an alarm if vehicle is tampered with.
- Operate on 10 degree slopes.
- Diesel primary motive power.
- Provide video, status, and command data to the main control station using a non-jammable, non-detectable communication link.
- Automatically avoid obstacle or prevent running into obstacles with a desired 100% assurance rate.
- Provide self-contained power capability for a minimum of eight hours continuous mobile operation.
- Be able to operate in an environment which contains fixed IDS sensors. Operate also in conjunction with the MDARS interior systems.
- Have a full teleoperation mode that will allow the operator to perform assessment and respond to threats.
- Be able to automatically query and update lock status on ammunition bunkers using a microwave detection system on a real-time basis.
- Be able to automatically collect inventory data of bunker contents using microwave tag collection system and compare to known inventory on a real-time basis.
- Be able to autonomously check the status of fixed barriers such as doors or fences.
- Have ability to detect exterior fires.
- Provide continuous video to the operator control station from all eight system for potential simultaneous recording and data collection.
- Provide bi-directional audio information.
- Be designed so that production cost in lots of 200 is approximately $150,000 per platform.

IV. Preliminary Design

Currently RST is involved in the preliminary design stage with. For design purposes, the system has been broken into seven different areas. These areas are:

- Navigation
- Obstacle Avoidance
- Intruder Detection System
- Lock/Inventory Monitoring
- Communication
- Vetronics/Platform
- Command and Control

• Our approach in all of these areas is to have a primary and secondary method to ensure mission success. Candidate solutions are discussed in the following sections.

Navigation

The navigation system will depend on two primary position approaches - a highly accurate low-cost Radio-Frequency (RF) locating system using 3 fixed base stations and vehicle dead reckoning. The output of both of the system will be fed into a Kalman filter to obtain an absolute position less than 0.15 meters over the six by six kilometer zone.

Using a infogeometric code division multiple access RF spread spectrum system, we will be able to obtain accurate position and bearing data - in essence a virtual navigation sensor. In order to ensure constant communications with the RF locating systems, we will operate on three simultaneous, redundant spread spectrum frequencies - 50 Khz, 1920 Khz, and 2400 Khz. This approach will make our system virtually unjammable and unbreakable with encrypted codes. Another advantage of the system is that every vehicle knows the position of every other vehicle at all times.

RST's software navigation methodology approach is to use position measurement data to provide position matching to a digital map. This map, in addition to terrain and path data, will contain location of expected landmarks, microwave tags, obstacles, and any other important items (this map will be used to control both navigation and detection/assessment behaviors of the system). This data will be incorporated using a combination of proven navigation software methodologies that has developed and demonstrated on the MDARS interior platform. These concepts include: Virtual Paths, Fuzzy Fit, and Event Driven Reentrant Behaviors.

Virtual paths is the division of routes into short, concisely defined, and easily modified path segments that are combined to form complete route programs which allow the vehicle to navigate between any two points in the system. Each path segment contains all the navigation, control, and personality data required to permit the robot to perform its mission along the segment.

Our unique fuzzy logic algorithms extend fuzzy logic to include the concept of two dimensional degrees-of-membership. Using these techniques, sensor inputs are automatically tested against a position estimation and confidence. Data is accepted (or believed) in proportion to its level of agreement. Using this technique, past sensor readings are automatically integrated with new data with the result being that the vehicle exhibits smooth, accurate, and purposeful control even in the presence of erratic navigation sensor data.

Under the virtual path approach, the vehicle attempts to close on and navigate along a precise path. To accomplish this, the vehicle must use event driven reentrant behaviors to change behaviors as the result of both expected and sometimes unexpected events. An example of an expected behavioral change would be the beginning of a turn to join smoothly with the next path segment. An example of an unexpected event would be the required circumnavigation of an obstacle.

Another recent navigation methodology development is a clean and simple technique that permits the vehicle to change behaviors while maintaining the context of each behavior for possible reentrance. For example, when a vehicle has finished a circumnavigation maneuver, it can return to the normal routine of collecting and processing navigation data without the loss of previous landmark information.

Obstacle Avoidance

Because of the high reliability required for the obstacle avoidance (OA) system, we are planning to use three different sensor methods. The first approach is a vision based system using a front facing stereo camera arrangement for object detection. The image processing of this data will be handled through time-sharing the same electronics designed for the Intrusion Detection System image processing (see next section).

The second method is the use of an array of ultrasonic sensors in the front of the vehicle. One low cost concept that we are currently exploring using three transmitting sensors and a single receiving sensors. With the proper signal processing, we will be able to derive a 3 dimensional acoustic image out to 10 meters with spatial resolution of around 3 inches. This approach would provide 100 degrees of horizontal and vertical coverage.

Finally we are examining several low cost radar systems that are currently on the market. Final selection will be made after a full evaluation on our remote controlled testbed.

Intruder Detection Systems

The primary IDS system will be vision based using a thermal imager (FLIR) and a pair of image intensified Gen 3 cameras specially arranged to reduce the motion parallax problem. Using a stop and stare technique with a rotating mirror, we believe that this configuration will allow the detection of both running and crawling intruders while the vehicle is moving. We will be looking for motion, color cues, thermal hot spots, shapes and the presence of object in clear areas. As the backup method, we are examining several concept including an unique pulsed radar technique and a scanning laser system.

David Sarnoff Labs has pioneered the development of pyramid (wavelet) technology for computer vision. The pyramid is a multi-resolution image representation that provides a framework for implementing fast algorithms for motion, stereo, and visual search tasks. The pyramid/wavelet representation also facilitates object recognition by isolating key features based on scale, orientation, and texture or spatial/temporal pattern characteristics. The use of pyramid technology can provide enhancements in system speed (or reductions in system size and cost) by factors of 1000 or more compared to conventional approaches. This technology makes it possible, for the first time, to build a sophisticated vision system for real time applications using modest hardware.

The video processing functions provided are summarized in Table I. The test system is designed to support multiple vision functions simultaneously. Most functions will be processed at full video rate, 30 frames per second. Stereo and motion vision processing functions share hardware modules so these functions will normally be performed in alternate frame times, each at 15 frames per second.

The prototype vision system consists of a set of custom processing modules mounted on approximately 6 VME boards. It will be housed in a box measuring roughly 15 by 15 by 10 inches (without power supply), and will consume roughly 120 watts of power. Both size and power will be reduced significantly in future implementations of this system.

The vision system is capable of processing data from three camera channels at once, and it can switch between cameras on a frame by frame basis. For example, the system might process the FLIR camera and two stereo cameras during one frame time, then switch to the three channels (RGB) of a single color camera the next frame time.

The vision system is organized in a parallel pipeline architecture. The processing modules are connected to a specially designed backplane that can transfer images along 32 separate pathways simultaneously. The vision functions (motion, stereo, etc.) are implemented as software programs that control the flow and processing of image data. The system includes three digital signal processing (DSP) units and a microprocessor for control and analysis. An external disk is used to store reference images and other data. A display module is provided to overlay graphic information on displayed video. This is used both in system development and in presentation of information to a human operator.

This vision system design contains the flexibility to upgrade or replace vision functions through modifications to the software programs and through the addition of new processing modules.

The design of the video processing system proposed for MDARS testing is based on a moving target indicator (MTI) system built by Sarnoff for the Army Mission Command and delivered in June, 1992. It was designed to detect and track moving targets from a moving platform. While still under test and evaluation at MICOM, this system has already proven remarkably capable and can detect even small or camouflaged targets while the camera is moving. The MTI system is a prototype built on two custom 9U VME boards. Total parts cost is roughly $12,000, and power consumption is 120 watts.

The MDARS vision system will be an improvement of the MTI design in four important respects. First, the system speed will be increased so that it can perform motion analysis at 30 frames per second (the MICOM MTI system processes 15 frames per second). Second, the processing modules will be modified slightly to support stereo as well as motion analysis. The same processing modules perform electronic image stabilization and registration to reference images . Third, further modest additions will be made to the processing capabilities to support the other vision function (e.g., color and texture). Finally, a new backplane will be added to support flexible data communications.

Table I Functions of the Video Processing System

Vision Functions	MDARS Task Served	Specifications
Motion Vision (stationary)	To detect moving objects while the MDARS vehicle is stationary and the cameras are panning. This is required for detecting intruders.	• can detect camouflaged objects • can detect small or distant objects (one or two pixels in size) • 30 frames/second
Stereo Vision	To determine the distance to objects and the orientation and shape of the road surface. This is required to determine distance to moving objects, to determine whether they are within a secured area, and to estimate their size. It is also required to detect objects or ditches in the path of the vehicle while driving off road (pursuit mode).	• 10 to 30 stereo frames per second • 1/10 pixel disparity precision
Registration to Reference Images	To detect minute-to-minute or day-to-day changes in a scene. This is required to detect intruders who stand still when cameras are directed towards them, and move between camera scans. It is also needed to monitor stored inventory for tampering or loss.	• images aligned to 1/20 pixel • compensates for errors in camera positioning • 30 frames/second
Electronic Image Stabilization	To compensate for erratic camera motion as the vehicle moves over rough terrain. This is required for human viewing in the teleoperation mode, and for computer vision to maintain frame to frame correspondence.	• compensate for image translation and rotation • 1/20 pixel precision • 30 frames/second
Pattern Vision	To determine object shape. This is required for discriminating between humans, vehicles, and animals when they are detected as moving objects in a scene.	• 10 frames/second
Landmark Recognition	To identify visible landmarks such as buildings, trees, poles. This provides data for refining estimates of the vehicle's position based on stored maps. It also guides the vehicle to standard observation points for observing inventory using reference images.	• accurate to 1 foot • less than 3 seconds per position update
Color	To classify objects based on color. This improves reliability of target detection and discrimination. It also improves the system's ability to detect obstacles in the road.	• generates a set of compact color maps • 30 frames/second
Texture	To detect irregular patterns in the road that may signify obstacles or a rough surface. This is required for driving on rough terrain.	• generates a set of compact texture maps • 30 frames/second
Detection on the Move	To detect intruders while the vehicle is moving. Primarily a software improvement over fixed motion detection	• can detect while vehicle is moving • 30 frames/second

Lock/Inventory Monitoring

The RST contract requires the MDARS -E system to interface to the U.S. Army developed RF secure lock system and the RF inventory tag system. Jointly, the MDARS interior and exterior program will develop a common database structure for cataloging and up-dating inventory as information is gathered in real-time.

Communications

The same RF system used for navigation location will also provide the transmission medium for video and command and status data. Each of the three frequency used will have the capability to transmit 256 kbits of information. The 50 Khz channel will be the primary data link due to its non-line-of-sight ability. The other two line of sight channels will be backup in case of jamming or other interference.

We will use real-time data compression techniques to reduce the black and white video image to under 256 kbits per second. Command and status data will be transmitted in a bi-directional 4800 baud channel. Audio data will be overlaid with the video during the compression process using multi-media technology.

Vetronics/Platform

Current plans are to build a hydrostatic driven six wheel, all wheel drive platform. This platform will be 84 inches long, 51 inches wide and 30 inches tall with a center of gravity that will allow it to operate on 40 degree sideslopes. Each wheel will have independent suspension for maximum rough terrain capability. A diesel engine driving a hydrostatic propulsion system offers several advantages over a convention mechanical drivetrain. These are:

- The diesel engine operates at a constant speed within its optimal power range. Because the speed is constant, it is easier to shock isolate the engine vibration and to reduce engine noise.

- Electronic vehicle control is only two wires to a flow control valve and two wire to the valve controlling the ackerman steering system.

- Individual wheel motors lowers the center of gravity and pushes weights to the outside edges of the platform, making it more stable on sideslopes. Conversion to an all electric drive is easy with the replacement of the hydraulic motors with electric motors if desired.

- Hydraulic components are proven technology, rugged, immune to dust and low cost.

The basic vehicle electronics will be VME based. Our design will use the Controller Area Network (CAN) local area network for communication between subsystems. Software programming will be initial done using the "C" language and VxWorks, with eventual conversion to ADA by the third year.

Command and Control

The command and control station is being developed under the interior MDARS program. This console control up to 8 interior systems, 8 exterior systems and interface with any fix sensor system.

Electronics in the control station will allow for data recording of status and video data from all 16 system simultaneously. This data will provide an historical record of events and will assist the operator in the manual assessment part of his job.

V. Conclusion

The program schedule defines three major milestones. In January, 1994, RST will demonstrate key technology components in a standalone fashion. In January of 1995, RST will demonstrate two fully working systems at our facility. During the last 12 months, we will install and test the systems at a military site. During this 12 month period we will also be allowed to modify systems hardware and software components if required. At the end of this period, a formal acceptance test will be used to validate the exterior MDARS concept.

6034

N94-30539

S/3-63

204136

P-5

AIAA-94-1189-CP

Task Automation in a Successful Industrial Telerobot*

Philip F. Spelt
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P. O. Box 2008
Oak Ridge, Tennessee 37831-6364

and

Sammy L. Jones
Remotec, Inc.
114 Union Valley Road
Oak Ridge, Tennessee 37830

Abstract

In this paper, we discuss cooperative work by Oak Ridge National Laboratory and Remotec®, Inc., to automate components of the operator's workload using Remotec's Andros telerobot, thereby providing an enhanced user interface which can be retrofit to existing fielded units as well as being incorporated into new production units. Remotec's Andros robots are presently used by numerous electric utilities to perform tasks in reactors where substantial exposure to radiation exists, as well as by the armed forces and numerous law enforcement agencies. The automation of task components, as well as the video graphics display of the robot's position in the environment, will enhance all tasks performed by these users, as well as enabling performance in terrain where the robots cannot presently perform due to lack of knowledge about, for instance, the degree of tilt of the robot. Enhanced performance of a successful industrial mobile robot leads to increased safety and efficiency of performance in hazardous environments. The addition of these capabilities will greatly enhance the utility of the robot, as well as its marketability.

Introduction

The robotic system described in this paper results from a cooperative effort by the Center for Engineering Systems Advanced Research (CESAR), at Oak Ridge National Laboratory (ORNL), and Remotec®, Inc., a company located in Oak Ridge, TN. CESAR, sponsored by the Engineering Sciences Program of the Department of Energy (DOE) Office of Basic Energy Sciences, represents a core long-term basic research program in intelligent machines. CESAR research includes studies in multiple cooperating robots, multi-sensor data analysis and fusion, control of mobile robots and manipulators, machine learning, and embedded high performance computing. With support from the DOE Office of Nuclear Energy, CESAR has been performing applied robotics research, systems integration, and has provided overall coordination and management of a consortium of four university research groups (Florida, Michigan, Tennessee, Texas) in a program aimed at robotics for advanced nuclear power stations and other hazardous environments.

Remotec is a world leader in research and development of remote robotic technology for hazardous operation in nuclear plants, police/military explosive ordnance disposal, and fire fighting. The company's family of robots have found a worldwide clientele. They are being used by several nuclear utility industries and national research laboratories to perform waste handling, surveillance, and surveying. This paper describes the addition of a system of sensors, encoders and the required computing power to integrate the information gleaned from these sensors to enhance the teleoperation of a successful industrial mobile robot. All hardware additions are performed in a manner which preserves the factory-designed resistance of the chassis to environmental contamination. Moreover, as will be described in detail below, the functional additions which enhance the teleoperation of this robot are done in a manner which preserves the original factory functionality. This is desirable because the retrofitting of an enhanced interface to existing robots should require as little additional training of already skilled operators as possible.

The Andros Mk VI Robot

The mobile platform of the ANDROS robot, shown in Figure 1, consists of six cleated tracks including a pair of main driving tracks. Separate motors to drive two pairs of auxiliary tracks: a pair of articulated front tracks, and an additional pair of articulated rear tracks. This unique design enables the robot to climb stairs and slopes, crawl over obstacles and ditches, make turns in tight spaces, raise the entire robot body, and maneuver over rough terrain with different surface conditions. The ANDROS manipulator arm has five degrees-of-freedom (DOF), with a 210 degree pivot range for both shoulder and elbow. An additional DOF is provided by a torso rotation joint, in addition to the platform mobility. This configuration allows the arm to occupy a minimum space for its home position while

----

*Research supported by the Office of Nuclear Energy, Office of Technology Support Programs, U.S. Department of Energy, under contract No., DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

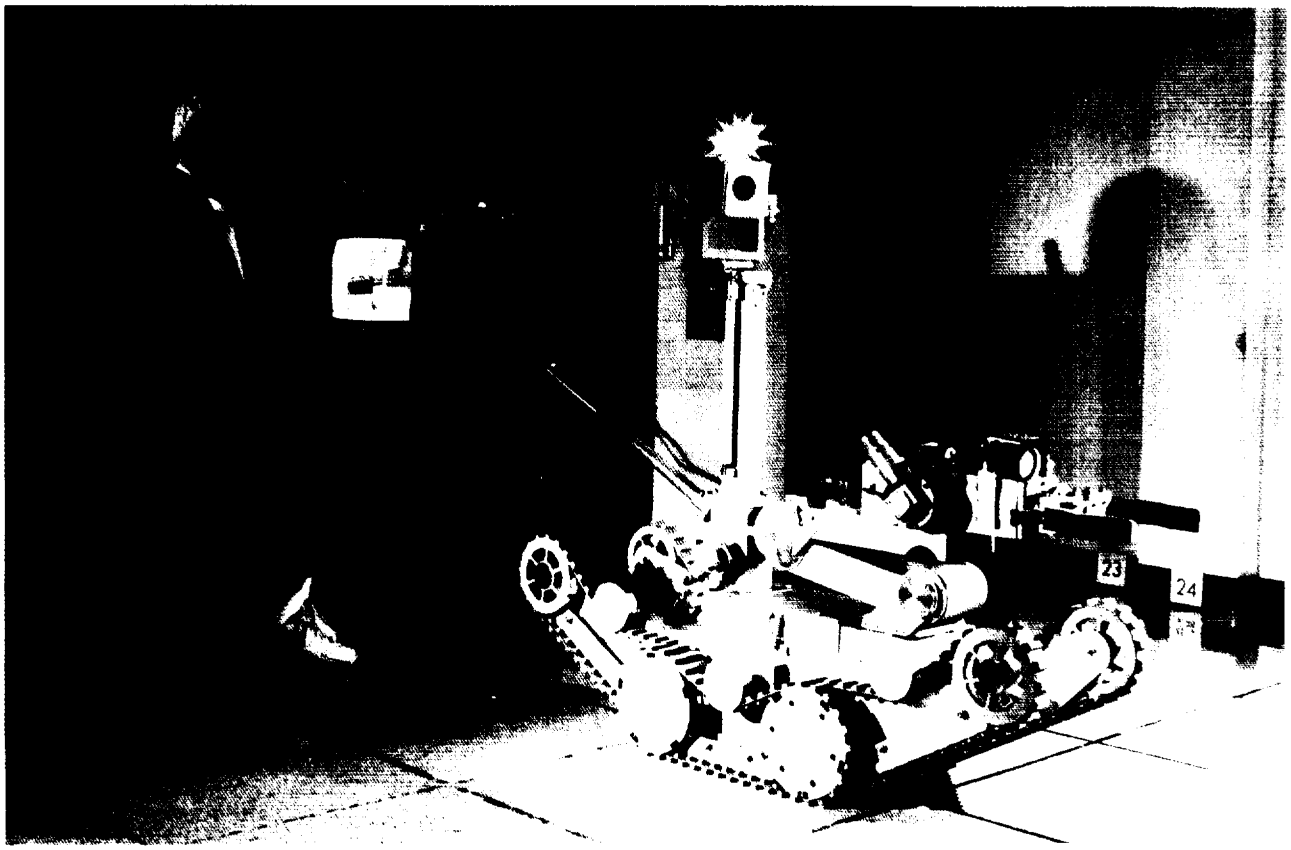


Figure 1. Andros Mk VI telerobot with control console in background.

providing maximum reach by folding down and extending straight out, respectively. Each joint is manually controlled with variable speed by individual switches on the control station. The wrist has pitch and six-inch extension capability, as well as continuous rotation, and the gripper has two parallel fingers controlled by servo-motors. The maximum lifting capacity is 40 kg.

The control station, shown in the background of Figure 1, consists of a switch pad with all the switches required to operate the ANDROS robot; a control console with a color television monitor, speaker, and microphone; and a console cable reel with a manual brake and hand-crank for the 100-m tether. Two video cameras are mounted aboard the chassis: a monochrome fixed-focus camera with automatic aperture is attached to the arm, and serves as a navigation camera when the arm is parked in the home position; there is also a color camera mounted on an extendible tower with pan, tilt, zoom, and focus capabilities under operator control. This camera serves as a general surveillance camera for both navigation and manipulator arm tasks.

In addition to the two-camera video feedback from the robot, two-way audio communication is available through a microphone/speaker system aboard the chassis and on the console. All told, there are 24 control functions on the control panel of the console, including the talk and volume switches for audio communication. Manipulating these control devices to smoothly control the robot and accomplish a task in the workplace requires considerable skill and practice on the part of the operator. In situations where the robot is out of direct sight of the operator, work must halt while the two cameras are used to assess current robot pose and the surrounding environment.

Workload considerations

Excessive workload on an operator of such a telerobot can degrade or slow down performance due to the number of task components which are manually performed. These components include manipulation of the cameras to monitor robot pose and tether placement, as well as to observe the effects of remote actions on the surrounding environment. In many cases, task performance must be interrupted to permit the operator to observe changes in robot pose as work progresses. The capacity to provide sensor feedback to the operator about robot position, articulator and arm position, and proximity of obstacles in the immediate environment, would greatly enhance overall performance of the system. In addition, automation of task components requires sensory feedback from the environment as well as encoder feedback about the positions of various robot components.

The procedure of automating a telerobot requires the addition of computer power to the robot, along with a variety of sensors and encoders to provide information about the robot's performance in and relationship to its environment. Custom software is required to integrate the encoder and sensor information and to use this information to provide automated control input to the

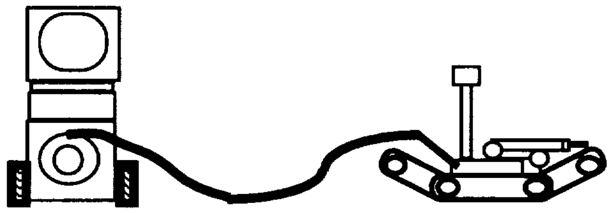


Fig. 2a. Factory configuration of Andros Mk VI robot and control console

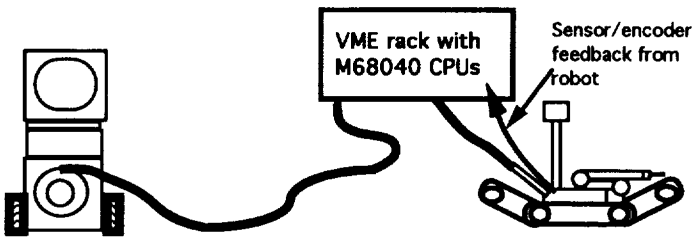


Fig. 2b. Additional computing power added to Andros Mk VI robot and control console

Figure 2. Illustration of original and enhanced Andros robot configuration.

robot. To be most effective, a variety of tasks must be automated, including obstacle detection and avoidance, planned manipulations by the arm and end-effector, and eye-gaze control of video camera pan and tilt. Addition of these capabilities will greatly enhance the teleoperation of an already successful industrial mobile robot. In order to accomplish these enhancements, a cooperative research and development agreement (CRADA) has been implemented between Remotec and ORNL. This CRADA involves equal inputs of time, effort and money on the part of both parties in order to create the enhanced robot described.

Enhancements to the Andros robot

As described above, the enhancements to the Andros robot require the addition of environmental sensors, encoders for the various robot movable parts, and computing power to provide the intelligence to integrate sensor and encoder information and provide automated control. The factory configuration uses an RS-232 digital data link (tethered or wireless) between the console processor and the onboard control processor. Analog control actions at the console are converted into digital signals and packaged and sent to the robot where they are decoded and converted back into analog signals to control the various motors on board. This design configuration permits relatively easy addition of computing power to integrate the added functions. As illustrated in Figure 2, the additional computing power is incorporated into the robot system by means of insertion into the RS-232 link.

The computing power added to the system is incorporated into a computer board cage (VME) in the form of two cards each containing a Motorola M68040 central processor unit (CPU) with associated memory and other necessary data processing devices. The cage is mounted on a custom-designed plate which attaches to the robot at the base of the pan/tilt camera tower is such a was that there is no permanent alteration to the configuration of the robot. This is desirable because the unit needs to be usable as a telerobot to perform tasks in contaminated areas which might arise during the course of this project. Therefore, one of the important goals of the CRADA is to be able to recover the original factory configuration of the robot, and to add the needed equipment in such a way that no permanent alterations are done which would, for example, reduce the contamination resistance of the unit.

One of the two added processors handles the incoming signals from the sensors and encoders aboard the robot. These data are processed through an analog-to-digital (A/D) signal converter prior to being sent to the first processor. This processor interprets and stores the incoming data, updating the data tables with new sensor and encoder information as required. The second CPU serves as a monitor of the control signals generated by the operator and sent along the RS-232 link. This unique arrangement permits this processor to either pass the control signals along unmodified or to alter them so as to modify the commands before they reach the control CPU in the robot. When the monitor CPU provides no signal modification, the robot operates exactly as the factory delivered it, in keeping with the CRADA goal of preserving the original factory specifications as a fall-back position.

Functioning of the enhanced control system

When the added control CPU functions to alter the control signals, it serves to move the robot from a totally teleoperated mobile robot in the direction of autonomy. Figure 3 depicts the now widely accepted situation in robotics in which high degrees of autonomy are attainable only in relatively simple tasks (the area under and to the left of the curve in Figure 3). The arrow pointing to the shaded oval in the upper right indicates the direction in which we are moving with the added computing power on the Andros. As more and more task components are automated, the robot becomes more fully autonomous. With the flexibility of the present

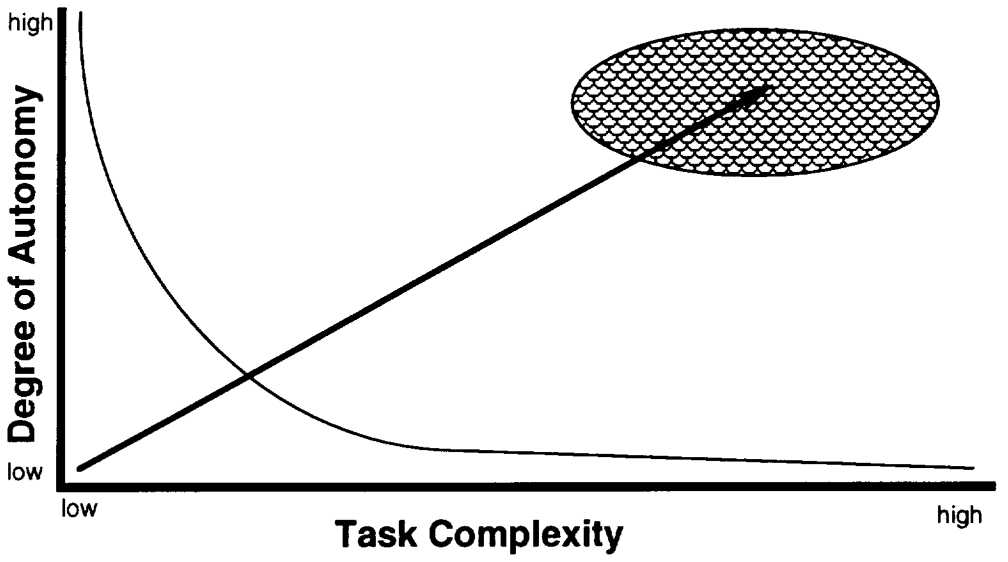


Figure 3. Diagram relating task complexity with degree of autonomy obtainable by most present-day robotic systems. The upper right oval represents the deisrable goal of high autonomy for very complex tasks.

system, different degrees of autonomy can be achieved as appropriate in different task environments.

Certain of the automated functions are planned to be permanent, while others may be invoked at some times and not at others. Many of the permanent functions fall into a class which can be designated as safety functions, and represent functions toward the lower left of the arrow in Figure 3. For example, the original robot is able to contact the pan/tilt camera tower with the manipulator arm, and it is the operator's responsibility to prevent this from occurring. With the enhanced control system in place, a software-derived envelope has been created around the camera tower, thus precluding accidental contact by the arm. Similarly, a variety of "illegal" configurations and poses can be defined which will protect both the robot and the environment from undesirable or dangerous situations. In this capacity, the CPU which monitors the control inputs simply changes the control commands to prevent the undesirable configuration from arising. This includes stopping the robot if it attempts to navigate a slope which is too steep in either pitch or roll, or if it is about to collide with an obstacle about which the operator is unaware.

Additional intelligent or automated capabilities serve to move the system toward the upper right along the arrow in Figure 3. At the simpler levels, these functions might include automated obstacle negotiation, manipulator or end effector tasks, and path planning. For example, a variety of repetitive manipulator tasks such as valve turning might be automated. In this case, the operator would position the robot so it could perform the valve closing, and the additional onboard CPU would assume the responsibility for actually closing the valve. At more complex levels of task automation (farther up and to the right in Figure 3), greater degrees of machine autonomy become involved, as more complex tasks are performed without operator intervention. This is one of the purposes of designing the enhanced operator interface for the Andros robot, and represents the type of new interface which will be fit to both existing and new examples of the robot line.

<u>Future research on operator-machine synergy</u>

In addition to serving as the testbed for developing the enhanced interface just discussed, this prototype system provides the opportunity to experiment with 'he advantages and disadvantages of varying degrees of task automation. These issues are of current interest in both aircraft cockpit automation and in the new designs of inherently safe nuclear reactor design (Spelt, 1993). Research in these areas indicates that operator boredom and takeover transients, when operator action is required, are a source of increased human error in highly automated systems.

Certain of the automated functions are planned to be permanent, while others may be invoked at some times and not at others. Many of the permanent functions fall into a class which can be designated as safety functions, and represent functions toward the lower left of the arrow in Figure 3. For example, the original robot is able to contact the pan/tilt camera tower with the manipulator arm, and it is the operator's responsibility to prevent this from occurring. With the enhanced control system in place, a software-derived envelope has been created around the camera tower, thus precluding accidental contact by the arm. Similarly, a variety of "illegal" configurations and poses can be defined which will protect both the robot and the environment from undesirable or dangerous situations. In this capacity, the CPU which monitors the control inputs simply changes the control commands to prevent the undesirable configuration from arising. This includes stopping the robot if it attempts to navigate a slope which is too steep in either pitch or roll, or if it is about to collide with an obstacle about which the operator is unaware.

Additional intelligent or automated capabilities serve to move the system toward the upper right along the arrow

in Figure 3. At the simpler levels, these functions might include automated obstacle negotiation, manipulator or end effector tasks, and path planning. For example, a variety of repetitive manipulator tasks such as valve turning might be automated. In this case, the operator would position the robot so it could perform the valve closing, and the additional onboard CPU would assume the responsibility for actually closing the valve. At more complex levels of task automation (farther up and to the right in Figure 3), greater degrees of machine autonomy become involved, as more complex tasks are performed without operator intervention. This is one of the purposes of designing the enhanced operator interface for the Andros robot, and represents the type of new interface which will be fit to both existing and new examples of the robot line.

Future research on operator-machine synergy

In addition to serving as the testbed for developing the enhanced interface just discussed, this prototype system provides the opportunity to experiment with the advantages and disadvantages of varying degrees of task automation. These issues are of current interest in both aircraft cockpit automation and in the new designs of inherently safe nuclear reactor design (Spelt, 1993). Research in these areas indicates that operator boredom and takeover transients, when operator action is required, are a source of increased human error in highly automated systems.

Ultimately, this system has the capability to perform complex tasks autonomously, using sensor-based feedback from the environment. As a result, this system will serve as a research vehicle for research into the manner in which automated task components can be seamlessly integrated with operator-performed components to yield a system which is capable of functioning in hazardous environments in a way which is both safer and more efficient than can be done under full teleoperation. Neither the manner nor the degree of task automation are intuitively obvious to observers of this process. Systematic research is required, in a variety of situations, to explore the most effective ways of capitalizing on the capabilities of both the human operator and the intelligent robot.

Andros robots are presently used by numerous electric utilities to perform tasks in reactors where substantial exposure to radiation exists. They are also used by the armed forces, as well as numerous law enforcement agencies. The automation of task components, as well as the video graphics display of the robot's position in the environment, will enhance all tasks performed by these users, as well as enabling performance in terrain where the robots cannot presently perform due to lack of knowledge about, for instance, the degree of tilt of the robot. Enhanced performance of a successful industrial mobile robot leads to increased safety and efficiency of performance in hazardous environments. The addition of these capabilities will greatly enhance the utility of the robot, as well as its marketability.

REFERENCE

Spelt, P. F. Role of the operator in nuclear power plants as determined from a survey of the North American nuclear community. Proceedings of the American Nuclear Society Topical Meeting, Oak Ridge, TN, April 10-14, 1993, pp 120-127.

→MG

Done
[illegible]

603 [illegible]

AIAA-94-1300-CP
N94-30540

CONTROLLING MULTIPLE SECURITY ROBOTS IN A WAREHOUSE ENVIRONMENT

H.R. Everett, G.A. Gilbreath, T.A. Heath-Pastore, R.T. Laird

Naval Command Control and Ocean Surveillance Center
RDT&E Division 531
San Diego, CA 92152-7383

[illegible]
P. 10

1.0 Abstract

The Naval Command Control and Ocean Surveillance Center (NCCOSC) has developed an architecture to provide coordinated control of multiple autonomous vehicles from a single host console. The Multiple Robot Host Architecture (MRHA) is a distributed multiprocessing system that can be expanded to accommodate as many as 32 robots. The initial application will employ eight Cybermotion K2A Navmaster robots configured as remote security platforms in support of the Mobile Detection Assessment and Response System (MDARS) Program. This paper discusses developmental testing of the MRHA in an operational warehouse environment, with two actual and four simulated robotic platforms.

2.0 Background

MDARS is a joint Army-Navy development effort intended to provide an automated intrusion detection and inventory assessment capability for use in DoD warehouses and storage sites. The program is managed by the Physical Security Equipment Management Office at Ft. Belvoir, VA. The Armament Research Development Engineering Center (ARDEC) at Picatinny Arsenal, NJ, has responsibility for inventory assessment and remote platform integration. The Belvoir Research Development and Engineering Center (BRDEC) at Ft. Belvoir, VA, is charged with security detection and assessment. NCCOSC is providing the command and control architecture and overall technical direction.

Reduction of manpower is a key factor in the MDARS cost benefit analysis. The objective is to field a supervised robotic security and inventory assessment system which basically runs itself until an unusual condition is encountered that requires human intervention. The host architecture must therefore be able to respond in realtime to a variety of exceptional events that may potentially involve several robots simultaneously. Distributed processing allows the command and control problems to be split among multiple resources, and facilitates later expansion via connection of additional processors.

2.1 Host Architecture Overview

A high-level block diagram of the MRHA is presented in figure 1. The number of Planner/Dispatcher and Operator Station modules resident on the host LAN can be varied in proportion to the number of deployed platforms at a given site. The initial prototype MRHA systems being developed by NCCOSC are configured with a Supervisor, two Planner/Dispatchers, a Product Assessment module, a Link Sever module, and one Operator Station for coordinated control of up to eight robotic patrol units.

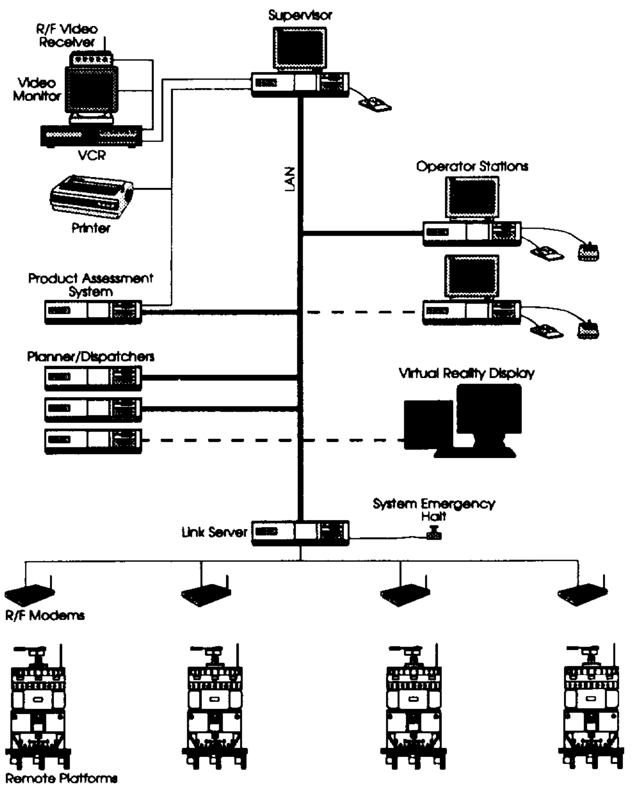


Figure 1. Multiple Robot Host Architecture (MRHA).

2.1.1 Supervisor

The heart of the MRHA is a 486-based industrial PC with a high-resolution display, referred to as the Supervisor. This module maintains a ready representation of the "big picture," scheduling and coordinating the actions of the various platforms, and displaying appropriate status and location information to the guard. Any hands-on control by the guard in response to situations requiring human awareness or intervention (i.e., alarm conditions, teleoperation) takes place at the Operator Station (see below).

Automatic assignment of resources (Planner/ Dispatcher, Operator Station) will be made by the Supervisor in response to exceptional conditions as they arise, based on the information contained in a special data structure that represents the detailed status of all platforms. Such exceptional conditions are referred to as *events*, and typically require either a Planner/Dispatcher, or both a Planner/Dispatcher and an Operator Station. Example *events* include: 1) an intrusion alarm, 2) a lost platform, 3) a failed diagnostic, and, 4) a low battery. The Supervisor will assign the highest priority need to the next available Planner/Dispatcher or Operator Station.

The Supervisor *Map Display Window* will automatically center on the platform listed at the top of the *Event Window*, which in essence represents the highest priority need. The guard can elect to split the screen and display up to four maps at once as shown in figure 2.

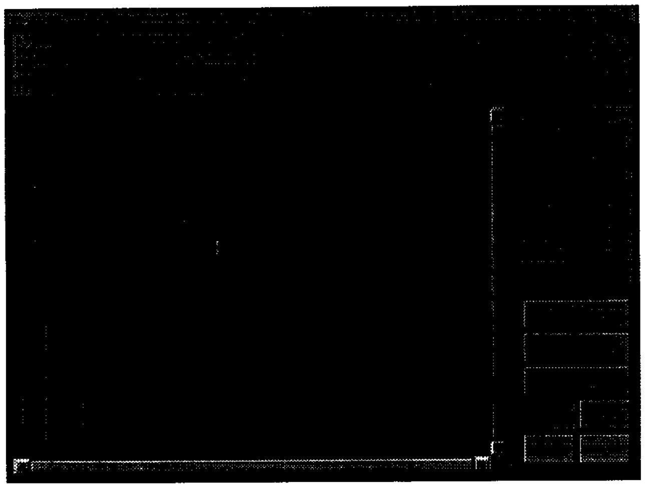


Figure 2. MRHA Supervisor Display.

2.1.2 Link Server

All the distributed resources within the host architecture communicate with the various remote platforms via an RF Link Server, which is interfaced to the LAN as shown in figure 1. This 386-based computer acts as a gateway between the LAN and a number of dedicated full-duplex spread-spectrum RF modems operating on non-interfering channels. The various resources (Supervisor, Planner/Dispatcher, Operator Station) on the LAN can thus simultaneously communicate as needed in realtime with their assigned remote platforms. In order to offload from these resources the tedium of constantly requesting status information from the individual remote platforms, the Link Server will periodically poll each platform for critical data such as battery voltage, position, heading, etc. This information is then stored in a blackboard for ready access.

2.1.3 Planner/Dispatcher

Referring again to figure 1, the Supervisor has at its disposal a number of Planner/Dispatcher modules linked over a high-speed Ethernet LAN. These 386-based PCs are mounted in a 19-inch rack adjacent to the display console as shown in figure 3. A globally-shared world model is maintained to provide a realtime collision avoidance capability complementing the Cybermotion *virtual path* navigation scheme employed on the K2A robotic platform. The Planner/Dispatcher modules perform the current virtual path planning functions of the Cybermotion Dispatcher (Holland, et al, 1990), and the *unrestricted path* planning functions of the NCCOSC Planner (Everett, et al, 1990).

The principal function of the Planner/Dispatcher is to plan a path (virtual or unrestricted) and download it to the assigned platform. Under normal conditions, virtual paths are executed until circumstances arise which require deviation from the pre-defined route segment. The most common example would involve an obstacle that blocks the virtual path, whereupon the *unrestricted path planner* is invoked to generate a collision avoidance maneuver.

2.1.4 Operator Station

The Supervisor also has access to one or more Operator Stations via the LAN. These modules are essentially individual control stations that can be

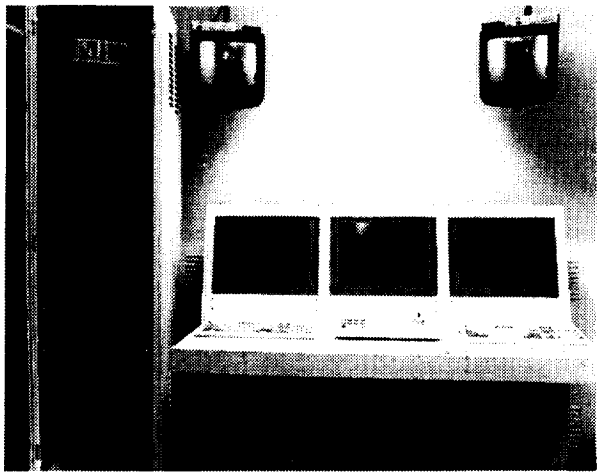


Figure 3. MDARS Control Console.

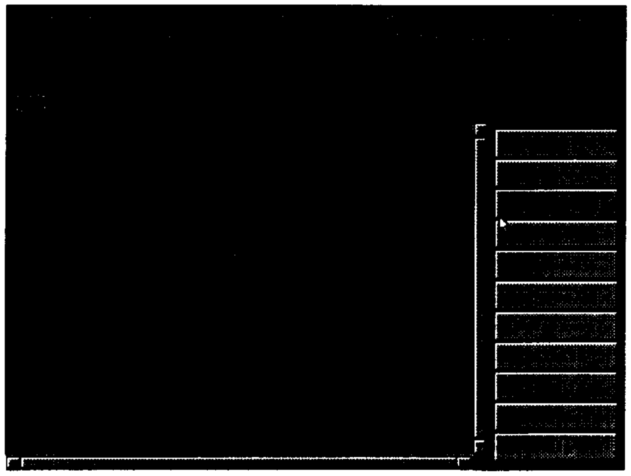


Figure 4. MRHA Operator Station Display.

assigned to a particular platform when the detailed attention of a guard is required. In this fashion, the Supervisor can allocate both computational resources and human resources to address the various situations which arise in the control of a number of remote platforms.

The Operator Station allows a security guard to directly influence the actions of an individual platform, with hands-on control of destination, movement, mode of operation, and camera functions. An Operator Station is automatically assigned by the Supervisor if an *exceptional event* occurs requiring human awareness or intervention. In addition, the guard can manually assign an Operator Station to: 1) teleoperate a platform when necessary, 2) perform non-automatic path planning operations (with the aid of a Planner), 3) place a platform in *Surveillance Mode* for intruder detection, 4) control an onboard video camera, and, 5) assess a potential disturbance.

The Supervisor and Operator Stations have been similarly configured to provide consistent, user-friendly visual displays. Both modules support a point-and-choose menu interface for guard-selectable options, commands, icons, and navigational waypoints. A row of command-option menu buttons are located on the right side of the Operator Station display screen as shown in figure 4. Telereflexive operation of the platform (Everett & Laird, 1990) and camera pan and tilt functions will be controlled by a specialized joystick.

2.1.5 Product Assessment System

The Product Assessment System is responsible for receiving actual inventory data from an interactive tag reader, and then correlating results with a database representing the supposed inventory. The robotic platforms are each equipped with a Savi tag interrogator that communicates with special RF transponder tags attached to the high-value or sensitive items to be monitored. The Savi tags respond with a unique identification code, which is then location-stamped and buffered in memory by the controlling microprocessors onboard the individual robots.

The buffer contents are periodically uploaded by the Link Server and passed via the host LAN to the Product Assessment System. The Product Assessment System compares each tag ID with information recorded in the database to determine if an item is mislocated or missing altogether. It also flags any detected tag IDs which are not represented in the database. A discrepancy report is generated at the end of each 24-hour shift.

2.2 Patrol Unit Overview

A block diagram of the platform architecture is presented in figure 5. Each robot is equipped with the Cybermotion SPI security sensor module providing full 360-degree intrusion detection coverage, augmented by a video motion detector. The high-resolution video surveillance camera is automatically positioned to view the scene of any suspected disturbance by a computer-controlled pan-

and-tilt unit, with live video relayed over a dedicated RF link to the guard console.

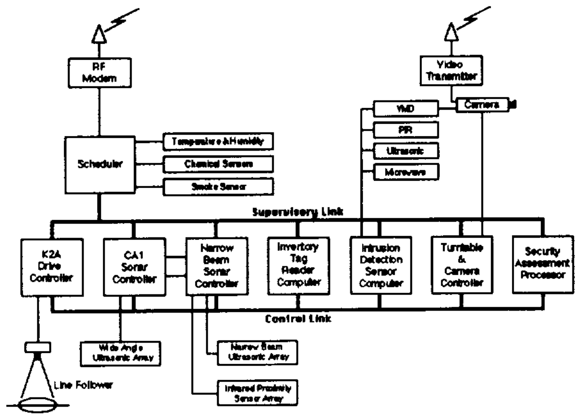

RF Modem
Video Transmitter
Camera
VMD
PIR
Ultrasonic
Microwave
Scheduler
Temperature & Humidity
Chemical Sensors
Smoke Sensor
Supervisory Link
K2A Drive Controller
CA1 Sonar Controller
Narrow Beam Sonar Controller
Inventory Tag Reader Computer
Intrusion Detection Sensor Computer
Turntable & Camera Controller
Security Assessment Processor
Control Link
Wide Angle Ultrasonic Array
Narrow Beam Ultrasonic Array
Infrared Proximity Sensor Array
Line Follower

Figure 5. Remote Platform Architecture.

An intelligent security assessment algorithm is employed to maximize the probability of detection while at the same time filtering out nuisance alarms. This onboard pre-processing of security-related data relieves the Supervisor of any security assessment responsibility, which is a key element of the control philosophy. The Supervisor basically gets involved only after an intruder has been detected and confirmed by the software onboard the robot.

To satisfy the immediate need for numerous remote platforms required to test the multiple robot control paradigm, a robot simulator was developed that implements the communications protocol of the Cybermotion K2A. The simulator hardware consists of a PC/AT-compatible laptop computer serially interfaced to an R/F modem of the type employed on the robots. The simulators are able to emulate specific platform functions, such as path downloading, decoding, and execution. During current development and test activities at Camp Elliott, four simulators are used in conjunction with the two Cybermotion K2A platforms.

3.0 The Navigational Problem

A wide variety of techniques have been developed over the years for the autonomous navigation of indoor vehicles. For purposes of this discussion, these may be grouped into three general categories: (1) guidepath following, (2) unrestricted path planning, and (3) virtual path navigation. Each of these guidance methods has advantages and disadvantages which determine its appropriate applications. MDARS seeks to integrate the desired features of all three techniques into a robust navigational package better able to cope with the varied demands of realworld operation.

3.1 Fixed Guidepaths

The simplest form of autonomous control involves a navigational control loop which reflexively reacts to the sensed position of an external guiding reference. Industrial vehicles have been guided by physical paths including slots, buried wires, optical stripes, and other methods for almost thirty years. Such automated guided vehicles (AGVs) have found extensive use in factories and warehouses for material transfer, in modern office scenarios for material and mail pickup and delivery, and in hospitals for delivery of meals and supplies to nursing stations.

The most common guidepath following schemes in use today involve some type of stripe or wire guidepath permanently installed on the floor of the operating area. Specialized sensors on the front of the platform are used to servo-control the steering mechanism, causing the vehicle to follow the intended route. These guidance schemes can be divided into three general categories: (1) those which sense and follow the AF or RF field from a closed-loop wire embedded in the floor, (2) those which sense and follow a magnetic tape in or on the floor, and, (3) those which optically sense and follow some type of stripe affixed to the floor surface.

The fundamental disadvantages of guidepath control are the cost of path installation and maintenance, and the subsequent lack of flexibility: a vehicle cannot be commanded to go to a new location unless the guidepath is first modified. This is a significant factor in the event of changes to product flow lines in assembly plants, or in the case of a security robot which must investigate a potential break-in at a designated remote location.

3.2 Unrestricted Paths

The term *unrestricted path planning* implies the ability of a free-roaming platform to travel anywhere so desired, subject to nominal considerations of terrain traversability. Most of the path planning work to date has been done on the premise that the ultimate navigation system would be capable of mapping out its environment with sensors, and then planning routes accordingly. While such systems have a great deal of appeal, they encounter several difficulties in practice.

The most significant problem associated with building a world model is the poor quality of most sensor data. There are many choices available to the designer of such a navigation system, but in every case good data is expensive. In practice, reflective sensors (ultrasonic rangefinders and near-infrared proximity detectors) have predominated (Everett, et al, 1992). Such sensors are subject to the problems of noise, specular and secondary reflections, and signal absorption to one extent or another. Furthermore, the perceived position of objects viewed from different locations will be distorted by any errors in the vehicle's dead reckoning accuracy as it moves between vantage points. Template matching of sensor data can thus be very difficult (Holland, et al, 1990).

Providing an autonomous capability to support non-restricted motion involves the implementation of an appropriate map representation, the acquisition of information regarding ranges and bearings to nearby objects, and the subsequent interpretation of that data in building and maintaining the world model.

3.2.1 Selecting a Map Representation

Several different map representation schemes have been devised, including polyhedral objects (Lozano-Perez, 1979), generalized cones (Brooks, 1983), certainty grids (Moravec, 1987), and quadtrees (Fryxell, 1988). The simplest scheme is a two-dimensional array of cells; each cell corresponds to a square of fixed size in the region being mapped. The map can be accessed and updated quickly, which is extremely important for realtime operation. Free space is indicated with a cell value of zero; a nonzero cell value denotes an obstruction.

The most compact form of a cell map consists of one bit per cell, and thus indicates only the presence or absence of an object. By using multiple bits per cell, additional descriptive information can be represented in the map, such as identification of structural walls and doorways. In addition, the probability of a given square being occupied can be easily encoded, which turns the map into a form of certainty grid (Moravec, 1987). This statistical approach is especially useful when the precise location of objects is unknown.

3.2.2 Unrestricted Path Planning Algorithm

A wide variety of path planning techniques have been developed over the years, each having various advantages and disadvantages. The actual planner employed in a given application is often dictated by which world modeling scheme has been chosen. For a certainty grid representation, the most straightforward planner is derived from the Lee maze router (Lee, 1961), with the cell coding enhancements suggested by Rubin (1974). The basic search algorithm begins by "expanding" the initial cell corresponding to the robot's current position in the floor map, (i.e., each unoccupied neighbor cell is added to the expansion list). Then each cell on the expansion list is expanded, the process continuing until the destination cell is placed on the expansion list, or the list becomes empty, in which case no path exists. This algorithm will find the minimum distance path from the source to the destination.

The minimal distance path, however, is not necessarily the "best" path. Sometimes it is more desirable to minimize the number of turns, or to maximize the distance from obstacles, for example. The search strategy can be altered accordingly by assigning a cost to each cell prior to adding it to the expansion list; only the minimum-cost cells are then expanded. This is known in the literature as an A* search (Winston, 1984), and was adopted by NCCOSC for use in this work (Everett, et al, 1990) due to the inherent flexibility of the associated cost function.

3.2.3 The Problem - Dead Reckoning Errors

Appropriate sensors must be coupled with some type of *world modeling* capability representing the relative/absolute locations of objects to support intelligent movement in unstructured environments. The accuracy of this model, which is refined in a continuous fashion as the platform moves about its workspace, is directly dependent upon the validity of the platform's perceived location and orientation. Accumulated dead-reckoning errors soon render the information entered into the model invalid since the absolute reference point for data acquired relative to the platform is incorrect. As the accuracy of the model degrades, the ability of the platform to successfully navigate and avoid collisions diminishes rapidly, until it fails altogether.

3.3 Virtual Paths

The virtual path concept was developed by Cybermotion to provide a routine mechanism for correcting dead reckoning errors in the normal course of path execution. Each desired route is pre-programmed by a technician to take advantage of any available environmental cues that the robot can recognize with its sensors. Each path begins and ends on named *virtual nodes* as shown in figure 6. A database is constructed that associates each virtual node with one or more *virtual path segments* entering or leaving that location. The Planner/Dispatcher uses this database to link several discrete virtual path segments together to form a complete route from any given node to any other node.

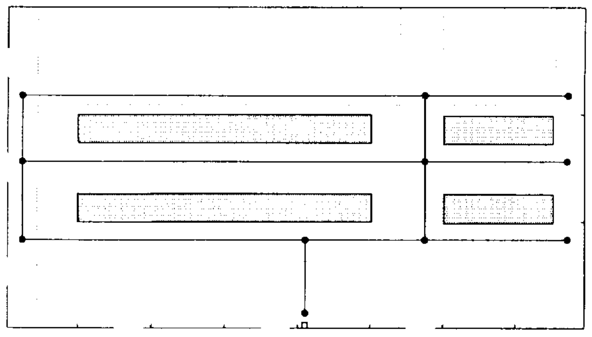


Figure 6. Map of Camp Elliott Showing Virtual Paths.

Correction of dead reckoning errors during run time is most commonly accomplished by indicating in the virtual path program (at the time of installation) one or both of the following cues: 1) the distance to the wall (or walls) on the left (and/or right) side of the robot, and, 2) the expected standoff from a wall in front of the robot at the completion of the route segment.

In the *wall-following* mode, the robot uses its lateral sonars to maintain the specified offset from the indicated wall while traversing the distance between two given points. Knowing the starting and ending locations of the virtual path segment, the robot can correct its heading as well as the lateral axis position coordinate. When approaching a wall, the robot uses the forward sonars to measure its actual distance from the wall. By comparing the observed range with the anticipated distance specified in the program, and knowing the X-Y coordinate of where it should be when it stops, the robot can correct the longitudinal axis of its dead-reckoned position. When *wall-following* and *wall-approach* are used together, both the X and Y coordinates can be corrected, in addition to heading.

Although the virtual path approach does not provide the flexibility of unrestricted path planning, it can be implemented with relatively low-cost sensor and computing hardware. Many practical applications can be addressed in this fashion, but the fundamental deficiency is the lack of collision avoidance capability. If an obstacle blocks a virtual path route segment, the platform must halt and wait for assistance.

3.4 MDARS Hybrid Navigation Scheme

The navigation scheme employed on MDARS is basically an integration of the Cybermotion *Dispatcher* and the NCCOSC *Planner*, which were separately employed on an earlier prototype to generate virtual paths and unrestricted paths, respectively. Integration of these two planning algorithms was accomplished in FY-92 under a Cooperative Research and Development Agreement with Cybermotion, giving rise to the term *Planner/Dispatcher.*

The hybrid navigational scheme exploits the inherent re-referencing ability of virtual paths, while retaining the free-roaming flexibility of unrestricted path planner control. Under normal conditions, the robotic platform traverses virtual paths, which are kept relatively obstacle free, at significantly higher speeds than typically possible in the unrestricted path mode. In the event of an impending collision, the platform is halted, and the unrestricted path planner generates an avoidance maneuver around the obstacle to the desired *virtual point* destination.

Three guidepath following strategies were investigated for incorporation into the hybrid navigation concept to support warehouse navigation. The location of a guidepath segment would simply be encoded into the virtual path database, the idea being to look down for a pathway instead of to either side for a wall. Finite path sections could then be strategically placed along troublesome route segments, as opposed to throughout the entire site.

The first of these path-following schemes was originally developed for AGVs by Litton Corporation, and employed a chemical stripe that glowed brightly when irradiated by an ultraviolet source. Disadvantages (from an MDARS perspective) included excessive power consumption, insufficient clearance between the floor and sensor, interference from ambient lighting, and periodic lamp failure. A alternative retro-reflective near-infrared design was developed by NCCOSC to overcome these concerns, but performance ultimately suffered from abrasive wear on the tape guidepath.

To address the problem of path degradation, a Hall-effect guidepath sensor developed by Apogee Robotics was purchased for evaluation, the intent being to bury the flexible magnetic tape in a saw kerf cut into the floor. This attempt proved futile as well due to limited (2-inch) sensor standoff, and the constantly changing magnetic signature of the K2A platform, which is an artifact of synchro-drive steering. As a consequence, the guidepath option has been indefinitely suspended until a practical solution compatible with the needs of a warehouse environment is found.

4.0 Warehouse Navigation

During the phased development of MDARS, three general classes of autonomous navigation are being addressed:

- Structured navigation - operation in a conventional walled office or laboratory environment.
- Semi-structured navigation - operation in a warehouse environment, with some structured order in the form of shelving (or inventory) forming permanent aisles.
- Unstructured navigation - operation in an open warehouse environment with no definitive aisles.

The MDARS cost-benefit analysis indicates the vast majority of operational sites visited to date fit the category of semi-structured, with fixed shelving but few or no unobstructed walls available for re-referencing. The remainder of this section discusses this class of navigation in the actual Camp Elliott warehouse environment. Camp Elliott is a government storage facility adjacent to Miramar Naval Air Station in San Diego, CA.

4.1 Random Patrols

The Supervisor automatically assigns idle platforms to a Planner/Dispatcher for random patrols of the warehouse environment. The Planner/Dispatcher generates a random virtual path patrol route, and downloads it via the Link Server to the assigned platform. This onboard K2A program contains instructions which cause the platform to halt and enter Surveillance Mode at randomly chosen virtual points along the path. When a platform arrives at its commanded destination, it reports back an *Idle Mode* status to the Supervisor. The Supervisor then reassigns a Planner/Dispatcher, which generates and downloads a new patrol route.

4.2 Obstacle Avoidance

Potential obstructions in the vicinity of the robot are detected by an array of Polaroid ultrasonic ranging sensors and Banner diffuse-mode near-infrared proximity sensors mounted on the front of the turret of the K2A platform as shown in figure 7.

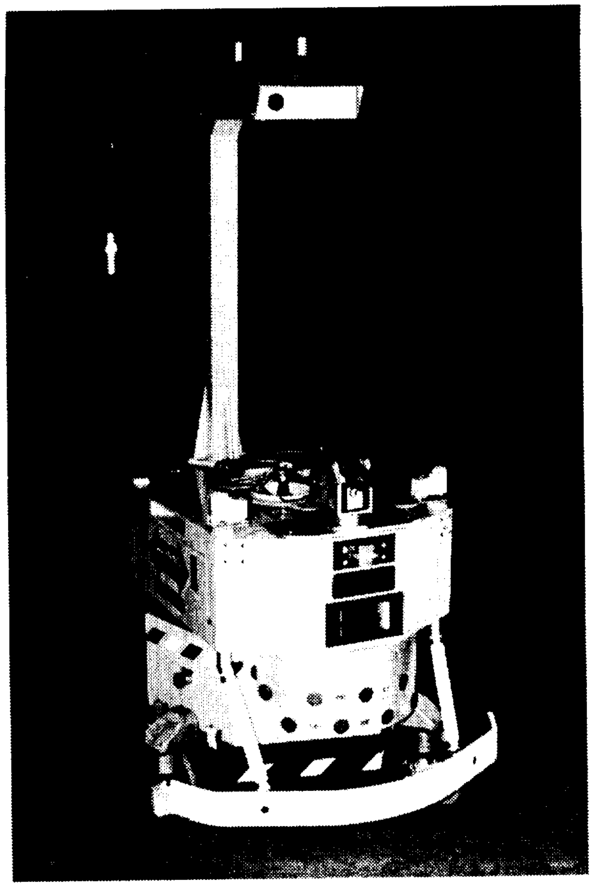


Figure 7. MDARS Remote Platform.

Range and bearing information collected over the last 10 feet of travel are stored in a circular buffer maintained by the Narrow-Beam Sonar Controller (figure 5). If a threatening object enters the protected envelope, the platform is halted with a *Blocked* status, which is in turn detected through routine polling by the Link Server.

Once the Supervisor has been made aware that a platform is blocked by an obstacle, it assigns a Planner/Dispatcher to resolve the problem. The historical sonar and proximity sensor data are uploaded from the platform after-the-fact via the Link Server, and used to update the world model for path planning purposes. The Planner/Dispatcher downloads the resultant avoidance maneuver to the platform for execution.

4.3 Navigational Re-Referencing

The big challenge posed by semi-structured warehouse navigation is nulling out accumulated dead-reckoning errors without any definitive walls. The wall-following and wall-approach instructions provided by Cybermotion are powerful enough in themselves to satisfy on-the-fly re-referencing needs in most structured environments. In a semi-structured warehouse environment walls are generally not available, but racks of storage shelves are usually abundant. If the inventory items do not significantly protrude over the lip of the shelf, the shelf itself can be treated as a wall and imaged by the side-looking sonars. Problems have been observed at Camp Elliott, however, in that some shelf sections are actually misaligned several degrees with respect to the path axis.

Experience has shown the primary source of accumulated position errors is erroneous heading information. During an extended dead-reckoning run, perceived position along the longitudinal path axis is usually quite good. Positional uncertainty perpendicular to the direction of motion, on the other hand, degrades much more rapidly due to minor inaccuracies in perceived heading. The problem manifests itself when the robot attempts to follow a rack or wall: if the measured range is too far (or too close), it will be treated as an anomalous sensor reading, and subsequently ignored. To compensate, an ultrasonic range "sniff" can be performed at the start of path segments where this problem is known to occur. This corrective action is accomplished by programming a very short *approach* instruction at the beginning of the virtual path, perpendicular to the direction of intended travel, on a distinctive target (i.e., a post, wall, or rack).

In most proposed depot locations, the MDARS robot will be patrolling multiple bays (typically 300' by 300') within the same warehouse. The robot traverses from one bay to the next through large fire doors that provide operational access for forktrucks and warehouse personnel. These definitive openings offer excellent on-the-fly opportunities to re-reference the X and Y axes of the robot through use of the Cybermotion *gate* instruction. Information is encoded into the virtual path telling the robot where to expect the doorway, the width of the opening, and path displacement from doorway centerline.

When approaching the programmed location, the robot begins to closely monitor the side sonars, looking for range readings that match the pre-specified characteristics of the gate. If a match is found, the vehicle's position along the length of the path is updated by substituting the longitudinal coordinate of the gate. Actual lateral position is determined by comparing the measured offset within the opening to what was expected. Unfortunately, however, no heading updates are directly obtained.

To augment the traditional *wall following*, *approach*, and *gate* instructions, an active re-referencing technique called *lateral post detection* has recently been incorporated. Short vertical strips of 1-inch retro-reflective tape are placed on various immobile objects (usually posts) on either side of a virtual path segment. The exact X-Y locations of these tape markers are encoded into the virtual path.

A pair of circularly-polarized Banner Q85VR3LP retro-reflective sensors are mounted on the turret of the K2A robot, facing outward as shown in figure 7. When the robot travels down a typical aisle as illustrated in figure 8, the Banner sensors respond to the presence of markers on either side, triggering a *snapshot* virtual path instruction that records the current side-sonar range values. The longitudinal position of the platform is updated to the known marker coordinate, while lateral position is inferred from the sonar data, assuming both conditions fall within specified tolerances.

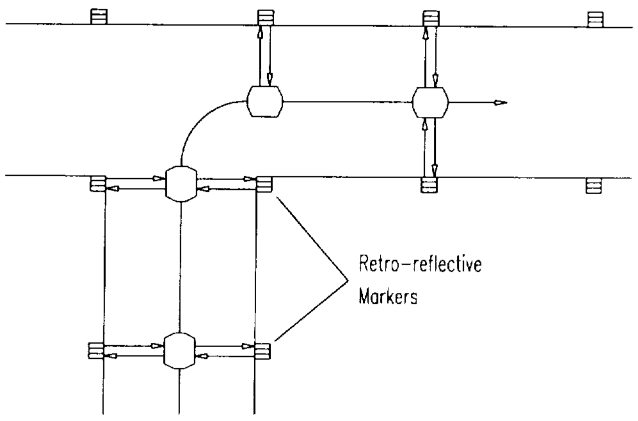


Figure 8. Lateral post detection referencing technique.

5.0 Camp Elliott Site Preparation

An 8' x 20' weatherized equipment shelter is used to house the host console electronics. The shelter is located adjacent to the warehouse as shown in figure 9, and is intended to represent a typical MDARS guard station.

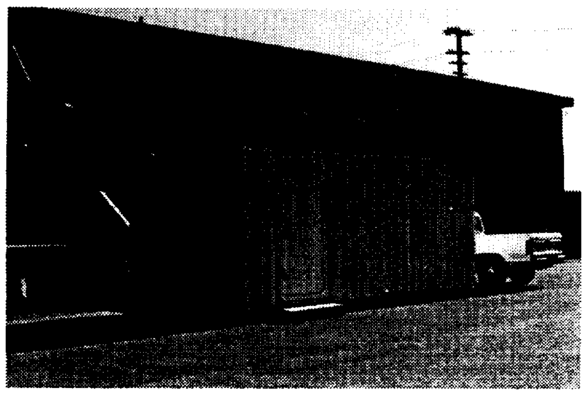


Figure 9. MDARS Camp Elliott control van.

An ARLAN 100-series R/F modem network is used to communicate simultaneously with multiple robots. The ARLAN 110 network controller connects to multiple daisy-chained ARLAN 010 transceivers located throughout the warehouse. Individual RS-232 serial channels are routed from the host console to the 110 controller within the warehouse.

If an exceptional event requiring human awareness or intervention should arise during non-attended operation, the Link Server will dial an outside line to alert a designated individual via appropriate speech output. This feature was implemented using a Versicom PROCAM development toolkit for creating custom automated telephone voice messages.

In order to respond to the automated operator intervention telephone calls, a remote control capability was implemented using the commercially available telecommunications package pcAnywhere by SYMANTEC. A high-speed modem is interfaced to each of the computers at the console via a code-activated switch. Users can dial-in and connect to any of the MRHA processors over the phone line and take control of the system just as though they were actually present at the remote site.

To aide in monitoring the movements of the robots from the host console, several CCD surveillance cameras were installed within the warehouse. The cameras are positioned strategically along and across aisles such that nearly the entire warehouse bay can be viewed. Since only two of eight video signals are returned to the console from the warehouse it is necessary to switch cameras in order to track the robots' movements.

The conical field-of-view for each camera is reduced to its projection on the X-Y floorplan, then further simplified to a rectangle to take into account restrictions imposed by the shelving on either side of the aisles. The positional coordinates of a pre-specified robot are repeatedly checked by the Link Server against each of these rectangular coverage areas to determine which camera holds the platform within its field-of-view. A digital command corresponding to the camera ID is then output over an RS-232 serial link to a one-of-eight video multiplexor, thus selecting the appropriate source for display on a monitor in the control van.

6.0 Status and Future Work

With the exception of the Product Assessment module, the hardware and software described in this paper is installed and functioning in the Camp Elliott warehouse facility. Savi tag reader hardware is currently being evaluated at ARDEC and should be installed at Elliott by March 1994; the first actual Product Assessment feasibility demonstration is scheduled for July 1994.

Version 1.1 of the MRHA was distributed to designated recipients in January of 1994, and several enhancements to the system are in progress or planned for the near future. These include automatic wall/shelf following during unrestricted planning operations, a scripting capability that allows more elaborate "canned" paths (useful for inventory monitoring), coordinated movement of multiple robots sharing the same operating environment, and improved methods of navigational referencing.

To enhance the effectiveness of the remote control capability described in section 5.0, a video capture and transmission feature will be added to the host console. This will allow video from one of several sources including the camera on-board the robot to be captured and transmitted to a remote user over the same phone line used for data and voice communications.

7.0 References

Brooks, R.A., "Solving the Find-Path Problem by Good Representation of Free Space," IEEE Trans. on System, Man, and Cybernetics, Vol. SMC-13, No. 3, 1983.

Everett, H.R., Gilbreath, G.A., Tran, T.T., Nieusma, J.M., "Modeling the Environment of a Mobile Security Robot," Technical Document 1835, Naval Command Control and Ocean Surveillance Center, San Diego, CA, 1990.

Everett, H.R., Laird, R.T., "Reflexive Teleoperated Control," Association for Unmanned Vehicles Symposium, Dayton, OH, Jul-Aug, 1990.

Everett, H.R., DeMuth, D., Stitz, H., "Survey of Collision Avoidance and Ranging Sensors for Mobile Robots," Technical Report 1194, Revision 1, Naval Command Control and Ocean Surveillance Center, San Diego, CA, December, 1992.

Holland, J., Everett, H.R., Gilbreath, G.A., "Hybrid Navigational Control Scheme for Autonomous Platforms," SPIE Mobile Robots V, 1990.

Fryxell, R.C., "Navigation Planning Using Quadtrees," SPIE Mobile Robots II, W.J. Wolfe, W.H. Chun, Eds., Proc. SPIE 852, pp. 256-261, 1988.

Lee, C.Y., "An Algorithm for Path Connections and its Applications," IRE Transactions on Electronic Computers, Vol. EC-10, September, pp. 346-365, 1961.

Lozano-Perez, T., and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," Communications of the ACM, Vol. 22, No. 10, pp.560-570, 1979.

Moravec, H.P., "Certainty Grids for Mobile Robots," Proceedings of the Workshop on Space Telerobotics, JPL, Pasadena, CA, January, 1987.

Rubin, F. "The Lee Path Connection Algorithm," IEEE Transactions on Computers, Vol. C-23, No. 9, September, pp. 907-914, 1974.

Winston, P.H., *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1984.

6036 AIAA-94-1190-CP

TECHNOLOGY TRANSFER AND EVALUATION FOR SPACE STATION TELEROBOTICS

N94-30541

Charles R. Price*
LeBarian Stokes
National Aeronautics and Space Administration Lyndon B. Johnson Space Center
Houston, Texas

Myron A. Diftler
Lockheed Engineering & Sciences Company
Houston, Texas

P. 9

Abstract

The international Space Station (SS) must take advantage of advanced telerobotics in order to maximize productivity and safety and to reduce maintenance costs. The Automation and Robotics Division at the NASA Lyndon B. Johnson Space Center (JSC) has designed, developed, and constructed the Automated Robotics Maintenance of Space Station (ARMSS) facility for the purpose of transferring and evaluating robotic technology that will reduce SS operation costs. Additionally, JSC has developed a process for expediting the transfer of technology from NASA research centers and evaluating these technologies in SS applications. Software and hardware systems developed at the research centers and NASA sponsored universities are currently being transferred to JSC and integrated into the ARMSS for flight crew personnel testing. These technologies will be assessed relative to the SS baseline, and after refinements, those technologies that provide significant performance improvements will be recommended as upgrades to the SS. Proximity sensors, vision algorithms, and manipulator controllers are among the systems scheduled for evaluation.

1. Introduction

The NASA Office of Advanced Concepts and Technology and the Office of Space Systems Development have sponsored and continue to sponsor the development of technologies that will improve SS efficiency and reduce life cycle operation cost. Technologies that expand the role of telerobotic maintenance and reduce the need for astronaut extravehicular activity (EVA) are particularly important and accordingly have been emphasized in NASA's overall telerobotics program plan [1]. Every hour of crew EVA time saved by using a robotic manipulator can be dedicated to the station's primary mission: scientific and engineering research. The use of telerobotic manipulators in this fashion is especially worthwhile, considering the high overhead in crew time required for each hour of EVA activity. In support of this task, the NASA JSC Automation and Robotics Division (A&RD) has established a technology transfer and evaluation process to determine which available technologies offer the most potential.

NASA JSC has a history of taking a leading role in transferring and evaluating telerobotic technologies in support of the SS program. JSC A&RD actively supports the integration and evaluation of the Canadian Space Station Remote Manipulator System (SSRMS) and special purpose dexterous manipulator (SPDM). JSC has supported extensive studies to determine the SS maintenance requirements at various points during the program's development [2]. In situations when new technologies are required as part of SS trade studies, A&RD has taken advantage of existing technologies developed outside the SS program. For example, the proximity detection and collision avoidance system implemented for an SS viewing study used a very fast distance calculation routine developed at the University of Michigan [3]. Also, a recent ground control study at JSC was built upon predictive display technology developed at the NASA Jet Propulsion Laboratory (JPL) [4].

JSC A&RD recently completed building the ARMSS facility for use in testing new telerobotic technologies. This facility provides a high-fidelity hardware SS environment for performing simulated maintenance activities. Previous SS maintenance activities simulated at JSC have been evaluated using fixed base manipulators to

*Chief, Robotics Systems Development Branch, Senior Member AIAA represent the Canadian SPDM operating in isolated SS work sites. The ARMSS facility goes well beyond this fixed base environment and any other existing NASA SS maintenance testbeds. The ARMSS testbed reproduces the relative motion that is possible between the SPDM base and its work site. In addition, its full scale SS preintegrated truss (PIT) segment provides realistic visual cues and obstacles for performing end-to-end maintenance tasks.

Over the years, NASA has invested in extensive basic robotic research and development at NASA centers and NASA sponsored universities. The work ranges from manipulator control systems designed at NASA JPL to sensors for robot collision avoidance developed at the Goddard Space Flight Center (GSFC). The university projects include a telerobotic protocol for Ethernet communications developed at Rice University [5] and fault tolerant manipulator concepts currently in work at the University of Texas [6]. These programs have provided prototype software and hardware systems with great potential for meeting SS productivity improvement goals. However, this potential may be achieved only if the prototype technologies are tested and refined in advanced applications development environments such as the ARMSS testbed and then, if they continue to show promise, further refined through flight experiments.

2. JSC Facilities

JSC maintains several robotic evaluation and integration facilities that provide support for SS. The Integrated Graphics Operations and Analysis Laboratory (IGOAL) supports non-real-time and real-time graphical simulation studies. IGOAL software is used extensively in determining the kinematic feasibility of many SS maintenance tasks. The Robotic Sensor Integration Laboratory (RSIL) provides support in the areas of sensor specification, design, and development and was used to refine the capaciflector sensor (described in a later section) provided by GSFC.

The Robotics System Evaluation Laboratory (RSEL) provides primary support for all SS tasks that require hardware simulation capability. The RSEL conducts qualifying tests using high fidelity robotic interfaces and orbital replaceable units (ORU's). Recent tests conducted in this laboratory with crew personnel were instrumental in the SS program decision to favorably consider ground control as a candidate for baseline operations [7]. Currently, prototype ORU's provided by the SS program are undergoing flight verification testing in the RSEL. In addition, the RSEL has provided the software and hardware tools that have been used to construct the latest JSC robotic laboratory addition, the ARMSS testbed.

ARMSS Testbed

The centerpiece of the JSC telerobotic technology evaluation facility is the ARMSS testbed. This 1-g simulator, shown in figure 1a, is NASA's highest fidelity SS maintenance environment for kinematic and contact tasks. It consists of three major components: an SPDM emulator, an SPDM mobility system, and a full scale SS PIT segment that together functionally reproduce the SS components shown graphically in figure 1b In this simulated view the SPDM is attached to the SSRMS and is preparing to replace an ORU located on a PIT door.

The ARMSS testbed can trace its origin to the previously designed but never constructed Automated Robotic Assembly of Space Station (ARASS) testbed for the on-orbit assembly of the 5-meter SS truss. After the SS change to PIT segments, emphasis shifted from dexterous robotic SS assembly tasks to SS maintenance tasks, and design work began on a telerobotic maintenance testbed. As much existing hardware as possible was incorporated from the earlier testbed into the ARMSS testbed, including the mobility system and the commercial manipulators. New hardware, such as the PIT segment and the ORU's, was designed and fabricated. An Intel Multibus II multiprocessor computer system, which is the SS standard, was purchased, and control system software written in "C" was transferred from existing JSC simulators to the Multibus II to serve as a starting point. To insure an operational system at the earliest possible date, all coding for the system was continued in "C." In keeping in line with SS requirements for eventual migration to Ada, an Ada compiler was purchased for the Multibus II, and a portion of the control system has been converted.

<u>SPDM Mobility.</u> The SPDM emulator hardware is mounted to a servo-controlled tower/rail system to achieve part of the mobility the actual SPDM will have when attached to the SSRMS. The system controller permits independent placement for each manipulator, both horizontally and vertically, and the manipulators may be positioned to achieve arbitrary SPDM placement and orientation within a 20-ft by 20-ft plane perpendicular

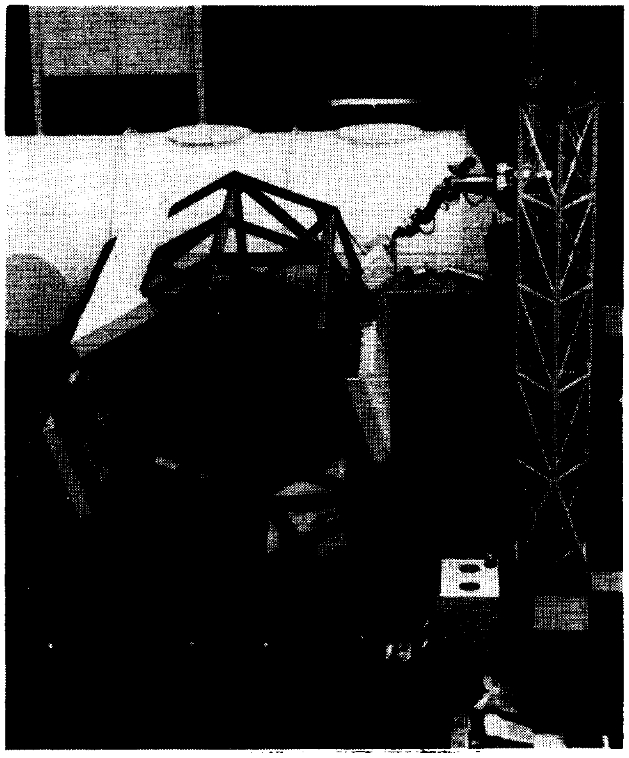


Figure 1a. ARMSS facility

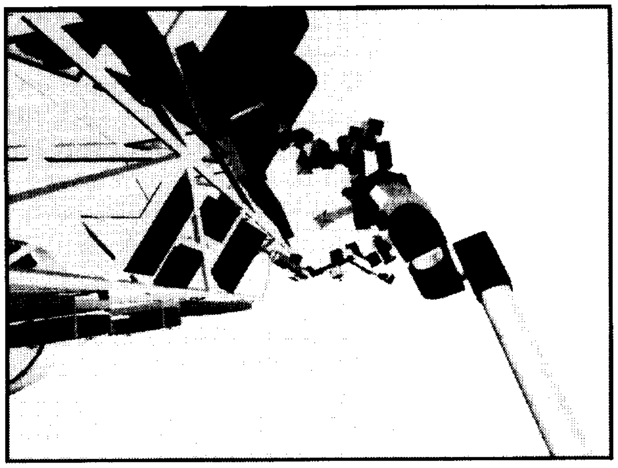


Figure 1b. Proposed SPDM attached to SSRMS

to the facility floor. The relative separation of the manipulators is adjustable to accommodate any future changes in the SPDM body design. Flexible cable trays run along the rails and towers providing power, data, and video communication to the manipulators.

The SPDM three-dimensional motion capability is completed with the addition of a mobile PIT structure. The PIT rests in a wheeled cradle that may be moved in a horizontal plane relative to the tower system. An actuator drives a high ratio gearbox, which in turn drives a chain and sprocket that rotates the PIT within its cradle about its long axis. Rotation is available in both directions. Combined, the tower and PIT degrees of freedom permit the manipulators full access to all six faces on the PIT.

SPDM Emulator. The SPDM emulator uses two commercially available Robotics Research (RR) 1607 manipulators (figure 2a) combined with the proper tooling to provide a very good approximation to the proposed SPDM arm (figure 2b). Like the SPDM manipulators, each RR1607 manipulator has seven degrees of freedom. The extra degree of freedom permits motion of the manipulator joints while maintaining a fixed end effector position and orientation. This is very useful in avoiding joint travel limits and obstacles, and reorienting the cameras that are mounted on the manipulator elbows At the time the ARMSS testbed was being designed, the RR1607 manipulators with tooling yielded approximately the same 2-meter reach that was planned for the SPDM. Subsequently, the SPDM design was modified, and now its arms are about 2.5 meters long. However, this is not expected to be a problem since extra travel is available by moving the ARMSS manipulator bases to increase or decrease the distance between the RR1607's. Finally, the RR1607 has sufficient capability to lift a functional 6B ORU, which is the one of the most common types planned for SS.

The tooling along each manipulator approximates the planned SPDM ORU tool changeout mechanism (OTCM) design. The ARMSS OTCM is shown in figure 3. After a manipulator is moved into proper position, the parallel jaw grippers located at the end of the OTCM grapple onto an ORU interface. Located directly behind and in between the gripper fingers is a shaft mounted socket. This device, known as the rotary drive, is extended after grappling and engages a bolt located in the center of the ORU interface. The rotary drive is designed both to loosen and tighten the ORU bolt. A force/torque sensor mounted behind the gripper connects the gripper to the manipulator and provides feedback for compliance control whenever the gripper is in contact with an interface.

PIT Segment and ORU's. The 20-ft PIT segment contains faces from two separate SS mission build (MB) segments, MB4 and MB2, and provides a comprehensive robotic maintenance

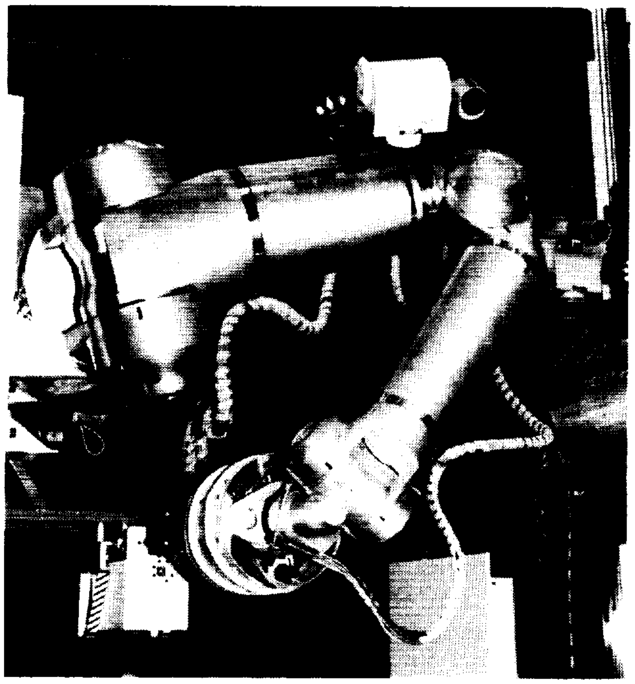


Figure 2a. RRK1607 manipulator

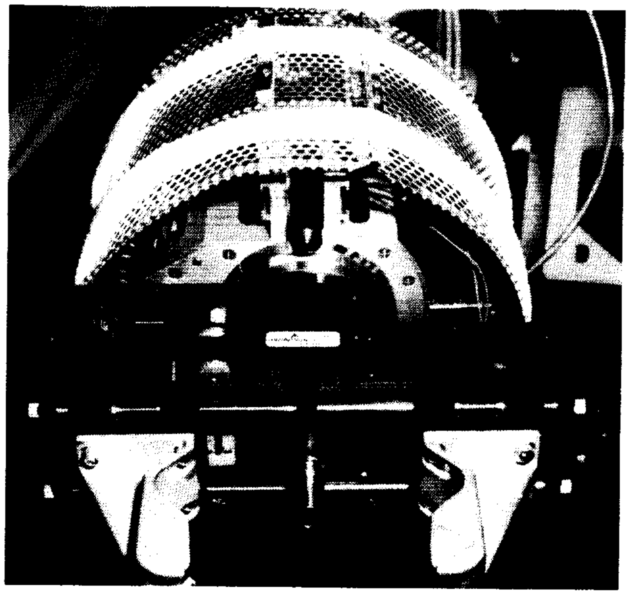


Figure 3. ARMSS OTCM

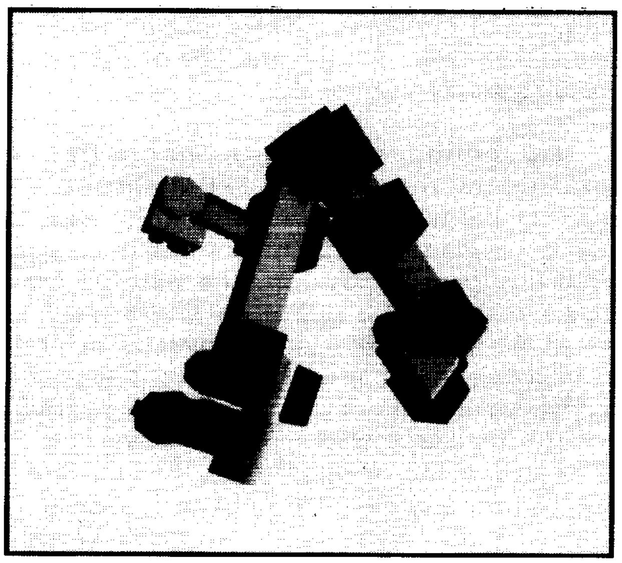


Figure 2b. Proposed SPDM ARM

testing environment. The structure is made of aluminum box section with a 4-in by 6-in cross section to match the SS PIT design. ORU doors are attached to two faces and contain attachment locations for both robot compatible 6B ORU's and various size noncontact 6B ORU mockups. The contact and noncontact ORU's may be rearranged to yield several possible configurations along the inside of doors. The SS program has designated that the doors will also be robot compatible, and the ARMSS PIT segment will be modified to accommodate the SS door design once it is released. Finally, utility trays are attached to several PIT sides providing realistic obstacles and viewing obstructions.

The ORU shown in figure 4 reproduces the functionality of one size 6B ORU. The interfaces for this ORU approximate the ones called for in the SS robotic system interface standards (RSIS) [8]. The manipulator grippers acquire the ORU by grappling the SPAR micro interface located on front of the box. A modified SPAR target located directly above the micro provides visual cues for manipulator alignment prior to grappling. The ORU is equipped with a box-to-cold-plate interface that slides into alignment guides located on the ORU carrier also shown in the figure. As the ORU is inserted and travels along the guides, it is pulled into position along the cold plate with the help of a manipulator force/torque compliance algorithm. The ORU is locked into place when the bolt located inside the micro engages the ORU carrier and is tightened down. An identical carrier may be mounted to any of nine locations on the PIT ORU doors.

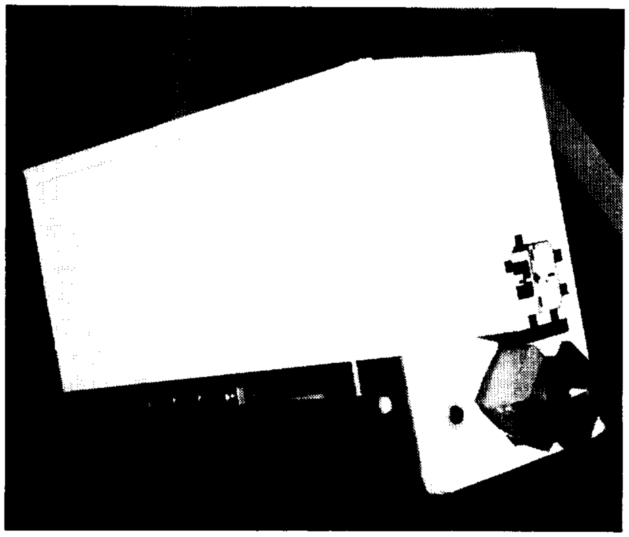


Figure 4. SS 6B ORU

Workstation and Video System. The ARMSS workstation shown in figure 5 reproduces the functional capability of the multipurpose applications console (MPAC) planned for SS. Two 486 personal computers (PC's) and one 386 PC run all the user interface and communications software. The ARMSS manipulators are controlled through SPDM displays that run on the upper left monitor and through two three-degree-of-freedom hand controllers located on either side of the workstation. A keyboard and a trackball are used to input data to the SPDM displays. Manipulator tooling is controlled through a combination of software display buttons and hardware switches located on the rotational hand controllers.

Two NTSC monitors, both with graphical overlay capability, provide video data to the operator. The video is controlled using either software displays or a push-button control panel on the upper right portion of the workstation. Both interfaces provide selection, pan and tilt, and zoom for each of the ARMSS cameras currently available. Referring back to figure 1, each manipulator has an end effector camera and an elbow camera. An SPDM head camera is approximated by a camera mounted just above one manipulator. A camera mounted to one tower simulates the elbow camera located on the SSRMS, which moves the SPDM from place to place. A single field camera may be moved as required to reproduce the capability of a relocatable SS PIT boom camera. All but the end effector cameras may be panned and tilted and zoomed, and each of these cameras has potentiometers for measuring pan and tilt position.

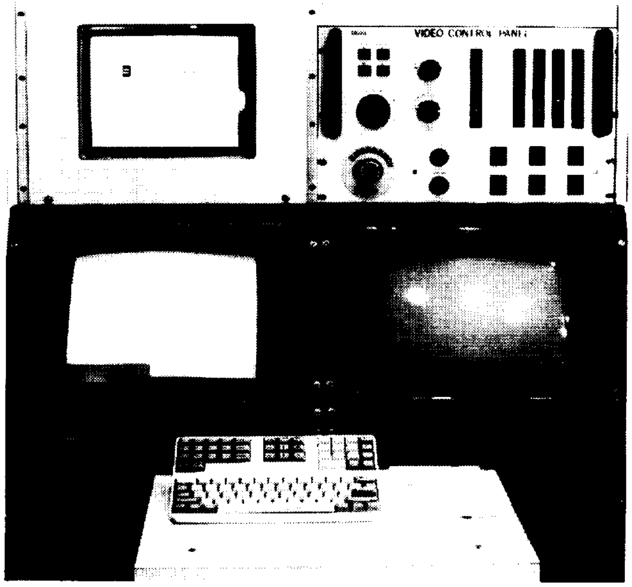


Figure 5. ARMSS MPAC workstation

This medium fidelity MPAC mockup uses the same interfaces for communicating with the manipulators and cameras that are specified for a high fidelity MPAC currently under development at JSC. The high fidelity MPAC replicates the video resolution and windowing capability specified for SS. It will also use the Sammi displays and controls software and the Lynx operating system planned for on-orbit operations. This high fidelity MPAC will be used to evaluate all SSRMS and SPDM displays as part of a separate JSC SS support project. The ARMSS workstation design did not incorporate these items due to a combination of cost and software maturity issues and only provides a subset of the SPDM control displays. However, when complete, the high fidelity MPAC will be interfaced with the ARMSS system and will be used to control the testbed when appropriate.

Control Architecture. The ARMSS control architecture outlined in figure 6 is based on the SS Multibus II standard. Separate processors are used to reproduce the relevant portions of the SS Mobile Servicing Systems Operations and Management Control Software (OMCS) and the SPDM control software. The OMCS processor receives commands from the ARMSS MPAC workstation, performs high level validity checks, and returns manipulator status back to the workstation. An Intel 386 20-MHz processor located in the same card cage runs the SPDM control system emulation software that communicates with the robotics research embedded processors.

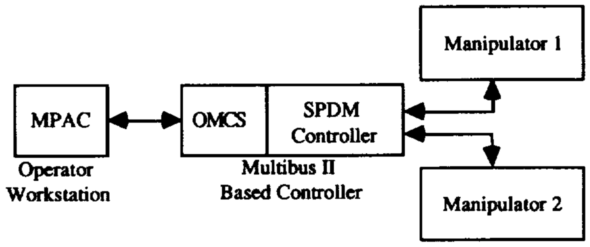


Figure 6. ARMSS control architecture

SPDM Control System Emulation. The Multibus II control software is designed to emulate all the SPDM kinematic and contact capabilities appropriate to a 1-g simulator. A detailed description of SPDM control software can and does fill several volumes; therefore, only the most important capabilities pertinent to SS maintenance are highlighted below.

The SPDM emulation software currently permits an operator to command each manipulator using the following baseline modes: end effector position and velocity, joint position and velocity, and pitch plane velocity. Prestored or operator position inputs are read by the SPDM software and converted to rate commands, which are sent to the RR embedded motion controller. The software constantly monitors a hand controller switch that must be engaged during operator commanded position moves. Motion may be stopped and restarted via this switch. Operator velocity command inputs are scaled by the software and limited to stay within SPDM specifications before being sent to the RR motion controller. Pitch plane, or null space, motion of the seven-degree-of-freedom manipulators commands are processed in a similar fashion. The SPDM emulation software is currently being expanded to queue up data for use in following prestored and ground commanded sequences.

One of the most important SPDM features available in the ARMSS software is force/torque compliance. The control software reads in data from the end-effector-mounted force/torque sensors and calculates commands to relieve contact forces that occur during ORU insertions and removals. The compliance commands are added to the hand controller commands, and the resulting "shared control" commands are sent to the manipulator. In addition, the emulation software provides a very simple form of gravity compensation by allowing an operator to bias out the force/torque sensor prior to initiating contact operations.

The SPDM emulation software provides coordinate transformations for both manipulator commands and feedback data. In addition, manipulator health is constantly monitored and provided back to the operator. Also, the ARMSS software commands the grippers and rotary drives used in ORU replacement operations based on SPDM specifications.

Communications and Ground Control Capability. Now that ground control of the SPDM is being favorably considered by the SS program as a viable means to reduce the maintenance burden for crew personnel [7], software and hardware that cannot be run on the SS due to power limitations will be considered for remote use. The ARMSS facility was designed with this requirement as a baseline and utilizes the TeleRobotic Interconnection Protocol (TelRIP) software [4], developed at Rice University, as the standard for communications with remote computers. TelRIP is a socket based data exchange mechanism, which allows multiple processes and processors to communicate in a common environment. Processes communicate through routers (TelRIP applications, which manage the flow of data between processes). Each application process contains a TelRIP stub, which maintains the socket connection with one router. Numerous, as well as remote, interconnections may be created over an Ethernet-TCP/IP network as multiple routers can maintain connections to each other as well as local processes.

All communications between the MPAC workstation and the Multibus II are run via TelRIP and include manipulator mode selection, hand controller inputs, tool operation, feedback data, and camera control. After incorporating TelRIP, ground based workstations such as the JPL Operator Control Station (described in a later section) have full access to the ARMSS testbed. The only exception is live video that cannot be accommodated along a shared Ethernet. If the operator control station is relocated to JSC, live video is readily available. In addition, TelRIP software routines developed at JSC to simulate ground-to-orbit data delays [3] and a separate PC based video delay system are available. For very high fidelity ground control simulations, a network has been established at JSC to route telemetry and video through the actual TDRSS system via a JSC communication station.

3. Technology Transfer Process

The technology transfer process includes three phases: coordination, implementation, and evaluation. During the coordination phase, JSC and a development center work together to identify candidate technologies that are suitable for SS applications. Once a technology is identified, a joint technology transfer plan is worked out with the contributing center, detailing the activities that each center will conduct to support the transfer.

Concurrence of this plan by both JSC and the development center signifies the beginning of the implementation phase. This is the longest phase of the process and involves the physical transfer of the technology to JSC. Supporting software is transferred to JSC, and JSC procures any specialized hardware required to host the technology. If appropriate, the technology is initially implemented in JSC RSEL using equipment that is compatible with the ARMSS architecture. This interim step is needed to reduce downtime on the high fidelity ARMSS testbed since it is most efficiently used as an evaluation facility with crew personnel as opposed to a debugging platform. The integration phase is completed when the JSC test coordinator and a representative from the development center agree that the transferred technology is performing properly.

Evaluation is the final phase in the transfer process. This phase begins with the completion of a test plan for the candidate technology. Test readiness reviews are held with the contributing center prior to test start. Tests are conducted to determine if the technology provides a performance improvement relative to the SS baseline. Representatives from crew training, mission operations, engineering, and the flight crew office perform controlled evaluations with and without the candidate technology. The evaluation phase is completed when the test report is produced. Technologies that provide performance enhancement are recommended to the SS program office.

The success of the technology transfer process hinges on choosing those technologies that will not only provide a performance enhancement to SS but also require minimal, or at most gradual, changes to SS hardware and software. For example, a new ORU grappling target that reduces operator workload and ORU changeout time would be installed on replacement ORU's. The target would be incorporated into future SS hardware replacements and not require a costly set of on-orbit replacements. The same is true for ground based telerobotic control software. Enhancements to a ground based system that do not affect the interface between the ground control center and SS would have a greater chance of acceptance than a control system that required additional onboard computing power. All candidate technologies for transfer are evaluated within this context.

4. Candidate Technologies

The following candidate technologies are either in the coordination or implementation phases of the transfer and evaluation process.

Flat Target

The first technology scheduled for transfer and evaluation using the ARMSS facility is the JPL flat target. This target is used as an ORU grappling aid and is viewed through a camera located on a manipulator end effector. As indicated by the name, the flat target is very thin. However, through the use of micro-lenslet array technology it produces a target that is projected approximately 1 inch from the face of an ORU. Thousands of quartz lenses that make up the target face produce this projected effect. The benefit is a low profile target that can be easily attached to an ORU and yet still provide three-dimensional alignment cues normally achieved with a much larger and heavier target.

The flat target is now in its third generation. Evaluations at JSC using the first two generations have provided useful feedback to JPL designers. The third generation target is expected to provide three times the resolution seen with the second generation. Using the ARMSS testbed, ORU changeouts will be performed with both the SS baseline target and with the flat target. Quantitative and qualitative test data will be collected, analyzed, and delivered to JPL for use in future refinements. This evaluation is expected to begin during the late summer of 1993.

Surface Inspection (SI)

The JPL SI system is a set of software routines for capturing and processing video images using a robot-mounted camera. An operator uses this system to drive a manipulator over a surface in either a manual or automated mode. The current surface views are compared to previously captured images using a video differencing algorithm. The system alerts the operator to any significant difference, and if appropriate, the operator logs the location of a flaw for future reference and/or repair. The SI system cancels out ambient lighting effects by using a set of controlled lamps that is also mounted on the manipulator. Images are processed under both ambient and a combination of ambient and controlled light. The images are subtracted to remove the ambient light effect. The user interface provides the operator with complete controls for all subsystems: manipulator, cameras, lighting, and image database.

The JPL SI system is scheduled for integration into the ARMSS facility during 1994 for testing under simulated SS lighting conditions. Crew personnel will evaluate the system in both manual and automated modes for inspection along the PIT mockup. Plans have been made to modify an ORU to show micrometeoroid damage. Significant work has already been completed with a partial transfer of the SI system to JSC RSEL. The software has been modified to work with the TelRIP communication system, and the SI user interface is currently being used to drive a JSC RR manipulator.

Capaciflector

The GSFC capaciflector is a device that measures the frequency of an oscillating electric field emanating from a flat antenna. As an object enters the field, it affects the permeability of the surrounding space and alters the oscillation frequency. The change in the frequency correlates to the distance between the antenna and the object. This device, also known as a capacitance proximity sensor, holds significant promise as an alignment aid for telerobotic ORU insertion.

A capaciflector prototype was transferred to JSC RSEL during 1992. After initial testing in JSC RSEL and consultations with the GSFC developers, a modified version of the sensor, which has greater thermal stability, was designed and developed at JSC. A graphic user interface that will provide short range proximity data to an operator is currently in work. After completion, the interface along with an ORU equipped with a set of capaciflectors will be integrated into the ARMSS control system. The benefits of the capaciflector versus the baseline ORU insertion alignment aids will be assessed during late 1993.

Operator Control Station (OCS)

The OCS developed at JPL is a prototype system for remotely controlling a manipulator system using saved sequences and intelligent macros. The system is designed for use when communication delays are several seconds long and direct teleoperation is not efficient. The OCS provides two main capabilities: a world model calibration system and a telerobotic control interface. The calibration system uses a combination of machine and human vision to accurately update the position of simulated objects and to build new ones on line. The telerobotic control interface is used to create and validate sequences in simulation before downloading to a manipulator. The sequences are stored in a convenient hierarchical fashion for use in executing entire tasks and may be easily modified by the user. The OCS was originally designed for interfacing with telerobotic devices located at JPL that have a higher level of autonomy than is currently baselined for the SPDM. However, much of this technology holds promise for use in ground control operations.

The OCS system has already been transferred to JSC and is currently undergoing integration with an RR manipulator. The system is being modified to use the TelRIP communications software, and additional handshaking is being incorporated to accommodate the SPDM baseline. Integration into the ARMSS facility is scheduled for late 1993. After an initial evaluation that adheres to the SPDM baseline capabilities, future testing that includes modifying the SPDM to include higher level capabilities or reflex actions will be planned.

HexEYE

The HexEYE proximity sensor is under development at the University of Southern California in conjunction with NASA JPL. The HexEYE is an optical-based proximity sensor that derives its name from the hexagonal configuration of its individual sensor units. This compact sensor has a footprint of approximately a square inch and provides distance data accurate to .3 millimeters within a 10-centimeter range. Ongoing refinements are expected to increase the range capability while still maintaining accuracy. HexEYE technology transfer activities are scheduled to start in 1994.

Exoskeleton

The JPL exoskeleton controller is an alternative to the planned SS hand controllers. This force-reflective exoskeleton fits around the arm and hand of a human operator and provides anthropomorphic manipulator control. This advanced controller will be incorporated into a ground control system during 1994 and will remotely drive an ARMSS manipulator. To use a force-reflective system in ground control, pseudoforces must be used to counter the effects of time delays in the communications loop. As part of the integration process, software will be developed to provide pseudoforces.

Remote-Site Robot Controller

The Langley Research Center is currently developing an advanced remote-site robot controller. This controller hosted on a manipulator local processor will provide a significantly higher level of automation than is currently planned for the SPDM. The intent is to elevate the operator to higher levels of supervisory control. It is expected that this system will complement the JPL OCS described above. The transfer and integration of this controller to the ARMSS facility is currently being planned.

5. Future Activities

NASA is continuing to invest in advanced telerobotic research and development activities in support of space exploration. Many of the generic technologies developed as part of this telerobotics program have the potential, when properly implemented, to improve SS productivity. In addition to the technologies already discussed above, current development activities throughout the NASA telerobotic community are being reviewed for technology applicable to SS. Work on fault tolerant robotic architectures at the University of Texas and icon based task control at Stanford University are among the technologies expected to be evaluated for SS in the future.

Acknowledgments

The authors would like to thank Andy Cordell, Dan Nolan, Phil Graves, John Schipper, Susan Shelton, David Wicks, Jeff Tassin, Reza Azad, Alan Bell, Lebarian Stokes, Richard Walker, Natasha Collmann, West Womack, Todd Sinkler, Thuy Nguyen, Thuy Ho, Jack Humphreys, Dirk Johnson, Bill Templeman, and Scott Askew of the JSC Automation and Robotics Division, and Steve Falls of the JSC Technical Services Division for their assistance during ARMSS system development.

References

1. Lavery, D. and Weisbin, C. "Telerobotics Program Plan." NASA Office of Advanced Concepts and Technology Document. January 1993.

2. Fisher, W. and Price, C. "Space Station Freedom External Maintenance Task Team Final Report," Vols. 1 and 2. NASA/JSC. July 1990.

3. Johnson, D; et al. "Proximity Detection and Collision Avoidance for Space Station Freedom Manipulators." Proceedings of the IEEE Annual Conference on Intelligent Robotic Systems for Space Exploration. Troy, NY. 1991.

4. Askew, R. S. and Diftler, M. A. "Ground Control Testbed for Space Station Freedom Manipulators." Paper No. 30. Proceedings of the IEEE Virtual Reality Annual International Symposium, Seattle, WA. 1993.

5. Wise, J. D. and Ciscon, L. "TelRIP Distributed Applications Environment Operators Manual." Technical Report No. 9103. George R. Brown School Of Engineering, Rice University, Houston, TX. 1992.

6. Tesar, D.; Sreevijayan, D.; and Price, C. "Four-Level Fault Tolerance in Manipulator Design for Space Operations." Proceedings of the First International Symposium on Measurement and Control in Robotics. Houston, TX. June 1990.

7. Space Station Level II Joint Program Review. NASA/CSA. January 1993.

8. Space Station Program Robotics Systems Integration Standards, Vol. II. NASA Document No. SSP30550, Vol. II, Draft. 1991.

5/6-63

201239

p 11

6037

AIAA-94-1191-CP

SPACE FLIGHT MANIPULATOR TECHNOLOGIES AND REQUIREMENTS FOR THE NASA FLIGHT TELEROBOTIC SERVICER (FTS)

N94-30542

John T. Chladek
NASA - Johnson Space Center
Houston, TX 77058

William M. Craver
Lockheed Engineering and Sciences Co.
2400 Nasa Road 1, Houston, TX 77058

Abstract

NASA Headquarters' Office of Advanced Concepts and Technology (OACT) joined efforts with Johnson Space Center's (JSC) Automation and Robotics Division and Langley Research Center's (LaRC) Information Systems Division to capture the technologies developed during the canceled NASA Flight Telerobotic Servicer (FTS) program planned for use on Space Station Freedom. The recent FTS Technology Capture effort completed the build and testing of one flight qualifiable FTS manipulator, deliverable to JSC's Automation & Robotics Division for environmental testing. The many robotic technologies developed to meet the 30 year space environment design requirements are discussed in this paper. The manipulator properties were to allow positioning control to one thousandths of an inch, with zero actuator backlash over a temperature range of -50 to +95 degrees C, and were to include impedance control and inertial de-coupling. Safety and reliability requirements are discussed that were developed to allow a thirty year life in space with minimum maintenance. The system had to meet the safety requirements for hazardous payloads for operation in the Shuttle Payload Bay during demonstration test flights prior to Station use. A brief description is contained on an Orbiter based robotic experiment and operational application using the dexterous FTS manipulator operating on the end of the Shuttle Remote Manipulator Systems (SRMS) from ground control.

Anticipated Mission Tasks

The original FTS concept for Space Station Freedom (SSF) was to provide telerobotic assistance to enhance crew activity and safety, and to reduce crew EVA (Extra Vehicular Activity) activity. The first flight of the FTS manipulator systems would demonstrate several candidate tasks and would verify manipulator performance parameters. These first flight tasks included

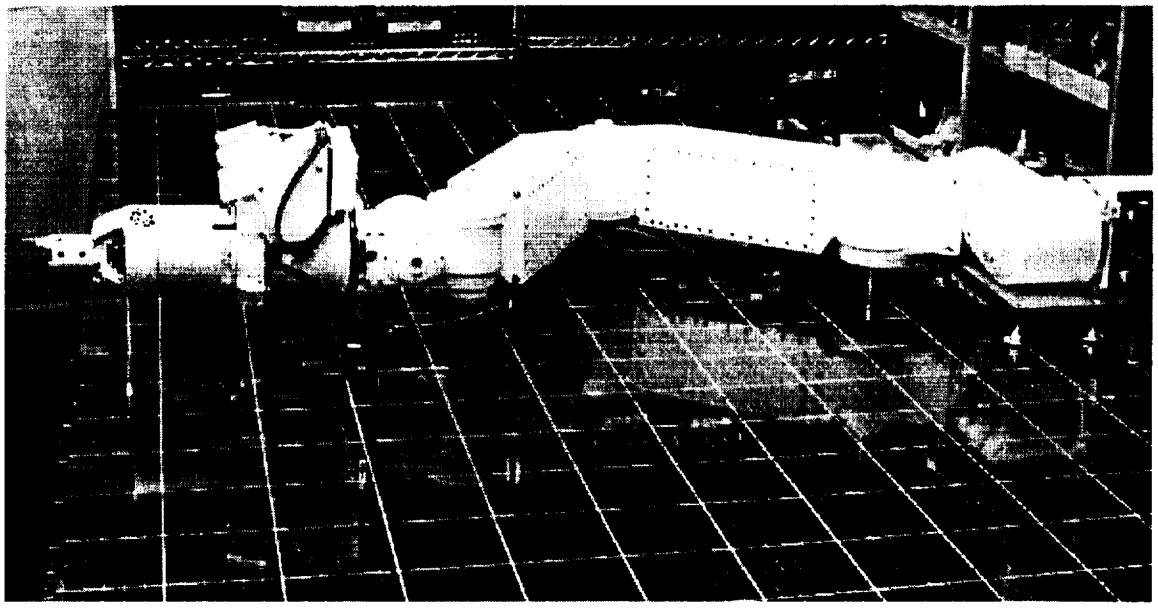


Figure 1 - FTS Manipulator on Air Bearing Table

unlocking a SSF Truss Joint, mating/de-mating a fluid coupling, contact following of a contour board, demonstrating peg-in-hole assembly, and grasping and moving a mass. Future tasks foreseen for the FTS system included ORU (Orbit Replaceable Unit) change-out, Hubble Space Telescope Servicing, Gamma Ray Observatory refueling, and several in-situ SSF servicing and maintenance tasks. Operation of the FTS was planned to evolve from teleoperation to fully autonomous execution of many tasks. The FTS manipulator has been assembled at Martin Marietta (see Figure 1) and will be delivered to NASA/JSC (Johnson Space Center). Successful component tests indicate a manipulator which achieves unprecedented performance specifications.

Currently anticipated tasks for dexterous space manipulators still focus on reducing EVAs as well as enhancing crew activity and safety. The Space Station (Freedom ?) plans to utilize a dexterous manipulator, SPDM (Special Purpose Dexterous Manipulator), on the SSRMS (Space Station Remote Manipulator System) to perform maintenance tasks such as replacement of ORUs. A potential being investigated for use on the Space Shuttle in assistance with EVA worksite setup and teardown. The first and last portion of most EVAs consist of placing or retrieving PFRs (Portable Foot Restraints), Tool Boards, and other devices needed to support EVA tasks. Other possible uses for dexterous manipulators include contingency use to avoid additional EVA crew intervention. The FTS manipulator requirements and designs are examples by which to assist in understanding current dexterous manipulator tasks and plans.

The wide range of FTS mission tasks combined with the desire to evolve toward full autonomy forced several extremely demanding requirements. Some of these requirements may be excessive to telerobotics community, but the FTS requirements appear to have been created to accommodate an open-ended evolution. This operational evolution would not be impeded by functional limitations in the FTS manipulator systems. Many of the FTS requirements discussed in the following sections greatly influenced the development cost and schedule of the FTS manipulator. A recommendation arising from the FTS program to remedy the possible impacts from such ambitious requirements is to better analyze candidate robotic tasks. Based on these task analyses, then weigh the operational impacts against development impacts prior to requirements definition.

Functional Requirements

The functional requirements of the FTS manipulator involve environmental, performance, safety, and resource effects. Many of these requirements are driven by the space environment, such as operation in thermal extremes, the need for safety, and limited resource availability (weight and power). Many of these requirements, however, focus on the manipulator and component functions to insure superior performance and ability to upgrade (evolution toward autonomy).

The primary robotic function of the FTS manipulator is that it move or manipulate objects in zero-gravity. Because interchangeable end-effectors were being considered, the manipulator requirements specify the tool-plate as the point of reference (see Figure 2 for FTS manipulator dimensions and components.) The tool plate is the attachment point for the wrist force/torque sensor. A manipulated object's mass may be as high as 37 slugs (1200 lb.) with the manipulator able to move masses less that 2.8 slugs (90 lb.) at velocities of 6 inch/second. Unloaded tool plate velocity will be at least 24 inch/second. Accuracy of tool-plate positioning relative to the manipulator base frame must be within 1 inch and ± 3 degrees. The manipulator must be able to resolve tool-plate incremental motion within 0.001 inch and 0.01 degrees. (Of coarse, verification tests of such extreme resolution specifications is costly.) Additionally, repeatability must be within 0.005 inch and ± 0.05 degrees with respect to the manipulator base frame. To perform useful work, the FTS manipulator was required to provide 20 pounds force and 20 foot-pounds torque output at the tool plate in any direction and in any manipulator configuration. These output force and positioning requirements were to be utilized with several control schemes including joint-by-joint, Cartesian, and impedance control.

To operate in space, the FTS manipulator had to meet the shuttle safety requirements as well as the environmental extremes. The safety requirements, as discussed later in this paper, ensure Orbiter and crew safety through fault tolerance. Safety is cited by Shattuck and Lowrie [1992] as "the single largest factor driving the system design." Safety and fault tolerance requirements resulted in monitoring of joint and Cartesian data, in checking of loop times to ensure proper functioning, in cross-strapping along communication paths, and in the addition of a hardwire control capability as a backup operational mode. Orbiter launch and landing impart vibration into the system which requires structural analysis and testing. Electromagnetic interference (EMI) must be limited both from invading and from exiting the manipulator systems. However, the most demanding aspect of the space environment from the FTS designer's view is the thermal vacuum of space. Operation in a hard vacuum (10^{-5} torr) and over temperatures from -50°C to 95°C, with directional heating and gradients, forced innovative designs, careful material selection, and extensive analysis.

Another consequence of the space environment is operation in zero-gravity. Designing the manipulator for a zero-g environment impacts structural, electromechanical, and electrical power considerations and well as the control system design. Because weight is a premium in space, motors are chosen to provide torque's for zero-g operation. This saves significant weight and electrical power when compared to motors chosen for ground-based operation. Smaller motors also benefit the thermal control system. The structure must also be lightweight, which increases flexibility and lowers structural bending mode frequencies. While being lightweight and more flexible, space manipulators are expected to handle payloads more massive than the manipulator. This expectation is far different from terrestrial manipulators which usually handle payloads less than 1/10 the manipulator weight. To maintain stability and performance of the FTS system, a 10:1 ratio is maintained between the first bending mode and the control bandwidth. This ratio precludes use of high bandwidth PID servos used in more massive, terrestrial manipulators. To address the stability and performance issues in the FTS manipulator, the structure was designed for stiffness (12 Hz first bending mode) and the manipulator control has a 1.2 Hz bandwidth, an inertia decoupler, and joint-level position, velocity, and torque servo control loops.

Manipulator Design Technologies

Beyond safety, FTS manipulator design was driven by the thermal environment and the positioning performance specifications. Of course, each manipulator subsystem was influenced by additional constraints and specifications. The following paragraphs describe the manipulator subsystem designs and technologies developed by Martin Marietta and its subcontractors to meet the FTS requirements and specifications. Manipulator subsystems discussed include manipulator kinematic design, link structure, actuators, control systems, and the end-of-arm tooling.

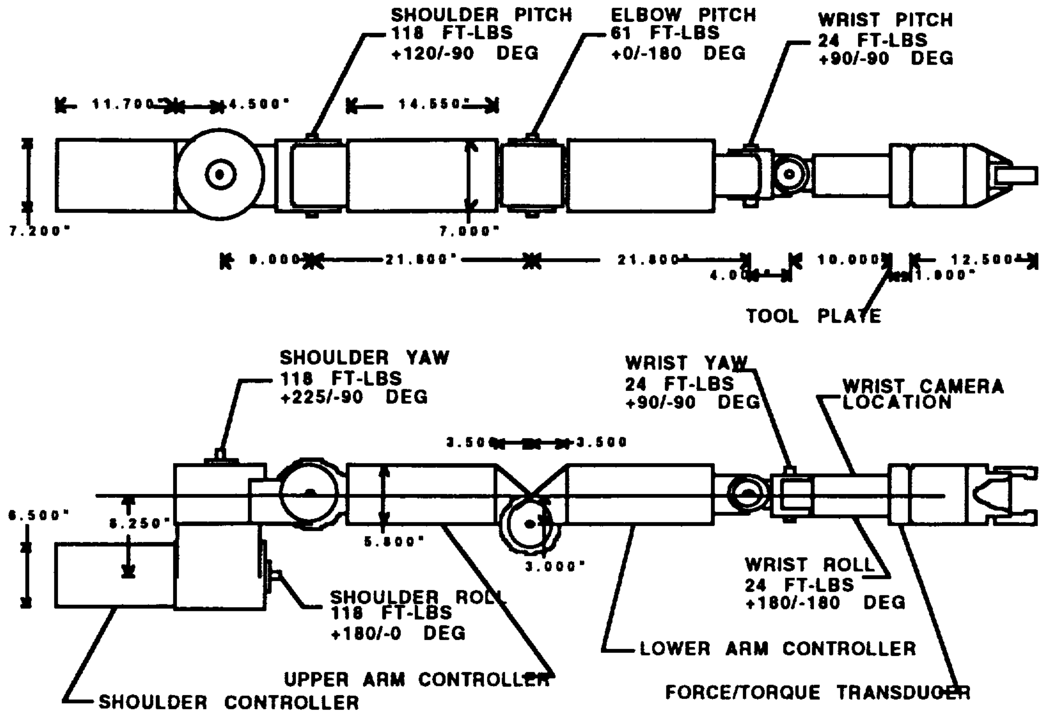


Figure 2 - Manipulator Dimensions & Components

Manipulator Kinematics

A 7-DOF (degree-of-freedom) R-Y-P-P-P-Y-R design is used with the first joint (shoulder roll) utilized for task-dependent configuration optimization. The outer 6 joints are actively controlled for coordinated output motion. The kinematic design has minimal joint offsets and 90° twist angles to simplify the kinematics. The 6-DOF kinematic arrangement, with three adjacent pitch joints, provides a closed-form inverse kinematic solution with few singularities within the manipulator workspace. The singularities which occur when the wrist roll or wrist yaw align with the shoulder yaw are beyond the usual workspace of the manipulator. Other singularities occurring at joint limits and when the elbow passes over the "home" position (see Figure 3), shown below, are eliminated with mechanical and software joint travel limits. The 3 inch displacement of the elbow joint is to allow the arm to fold back on itself for a greater workspace.

Link Structure

The manipulator links provide structural support as well as joint controller electronics packaging and thermal control. Packaging and thermal control determined link sizes while fracture and stiffness considerations drove the structural design of the links. A stiffness requirement of 1,000,000 pounds/foot and 1,000,000 foot-pounds/radian resulted in a smallest structural safety margin which exceeds 14, far greater than Shuttle requirement for a 1.4 factor of safety. Easy access to electronics is through side plates on the links. To avoid the cost and complication of active cooling, radiation is the primary thermal path. The controller boards sit in slots within the links which provide conduction paths to the link structure for radiation to the environment. Figure 4 shows the links and the computer cards which fit within the links. The link designs use material coatings, mounting hardware, and Kapton/Inconel film heaters to maintain thermal control.

Actuators

The joint actuator designs, developed by Martin Marietta and Schaeffer Magnetics, were also driven by positioning, performance, and thermal demands. These high-performance, zero backlash actuators each house a DC-motor, harmonic drive transmission, output torque sensor, output position sensor, fail-safe brake, hard-stops, and internally routed cabling. The design achieves considerable commonalty between actuators. Three sizes are used - one for the 3 shoulder joints, one elbow joint, and one for the 3 wrist joints.

The DC-motors have brushless, delta-wound stators with samarium cobalt rotors. This design offers good thermal properties, low EMI, minimal rotational losses, and linear torque-speed relationships. Motor commutation signals are generated from Hall Effect sensors, a second set of which is installed for redundancy. A secondary set of windings within the stator, driven via an independent electrical path, provides at least 10% rated torque and 0.5 degrees/second joint velocity for operation of a backup mode. This degraded mode of operation, commanded joint-by-joint satisfies the need for safing the manipulator after failure of a primary system. Fail-safe brakes attached to the motor rotor shaft are spring-loaded so that loss of power engages the brake. These brakes may be released with an EVA release bolt, which when turned 90° releases a cam on the brake armature.

Harmonic drives provide 100:1 backdrivable gear reduction in a compact volume. The harmonic drives were chosen for torsional stiffness and zero backlash. Cup size is determined by joint torsional stiffness requirements. In fact, because of the relative flexibility of the harmonic drive, all other torsion members are considered rigid. Rather than the standard Oldham coupling to the wave generator, a specially designed cylindrical coupler was used to eliminate backlash. Additionally, the output is coupled to a flange around the motor and harmonic drive. This flange, mounted to large duplex bearings provides compactness, rigidity, and an efficient load path to the output link.

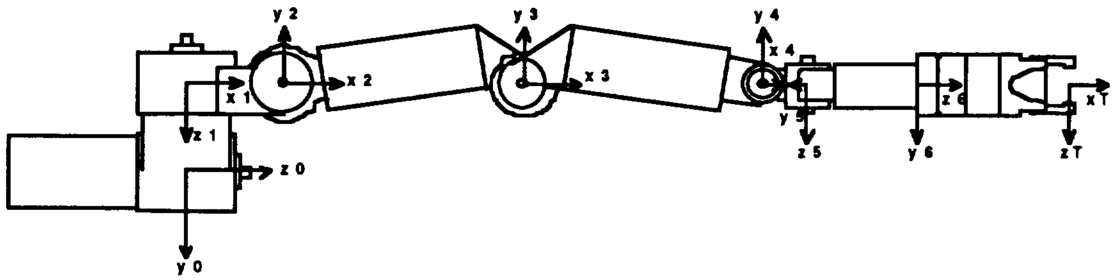


Figure 3 - FTS Manipulator "Home" Position

An analog torque loop is implemented in the joint servos to accommodate the non-linear and high-frequency affects of the harmonic drives. Sensed torque values come from an output torque sensor embedded on the harmonic drive output flange. Strain gages are mounted to the spokes of the titanium flange. This sensor placement isolates the sensor from structural loads (bending), thus primarily transmitting actuator torque. For effective performance, this analog torque loop operates at 1500 Hz.

Like the manipulator structure, the actuator housings and bearings were designed for stiffness and thermal stability. A standard bearing steel, 440C stainless, is used for all bearings. Bearing lubricant is Braycote 601, a liquid lubricant used in space applications. Its very low vapor pressure allows the actuator to be vented rather that sealed, but was still designed to resist contamination and assembly in a clean room. The motor bearings are deep-groove roller bearings sized for the thrust load of brake engagement and spring pre-loaded to minimize temperature sensitivity. The output bearings are large diameter, duplex-pair, angular contact bearings (face-to-face mounting). These bearings share radial and thrust loads with another duplex-pair on the other side of the actuator. An exception is the wrist roll, which has a single, duplex pair mounted back-to-back for better rigidity against the bending moments of the full cantilever load. Unfortunately, this back-to-back installation has greater sensitivity to assembly misalignments. This sensitivity may contribute to the excessive, uncompensated friction discovered during recent wrist roll torque loop tests.

The actuator housings are aluminum and titanium. Titanium is utilized near bearings. The similar thermal properties of 440C stainless and 6Al-4V titanium minimize temperature effects on bearing pre-loads. These pre-loads were determined as a compromise between stiffness and friction drag. The actuator case was designed for thermal needs. Motor and brake heat is dissipated to the ends or to the casing and then radiated to the environment. Like the links, the actuator design uses thermal isolation, material coatings, and internally mounted film heaters to protect bearings from thermal gradients. These gradients could adversely affect actuator friction and positioning accuracy.

The positioning and incremental motion requirements call for encoder data within an arc-minute which required position resolutions to 22-bits. To meet this need, inductive encoders were developed specifically for the FTS program by Aerospace Controls Corporation. These encoders have a fine and a coarse track used for incremental and absolute position resolution, respectively. Temperature effects on sensor accuracy were discovered during thermal testing. These errors were stable and repeatable with temperature, and are thus correctable in software.

All cabling in the manipulator is internally routed through the links and actuators. Each actuator has a cable passageway designed to eliminate twisting

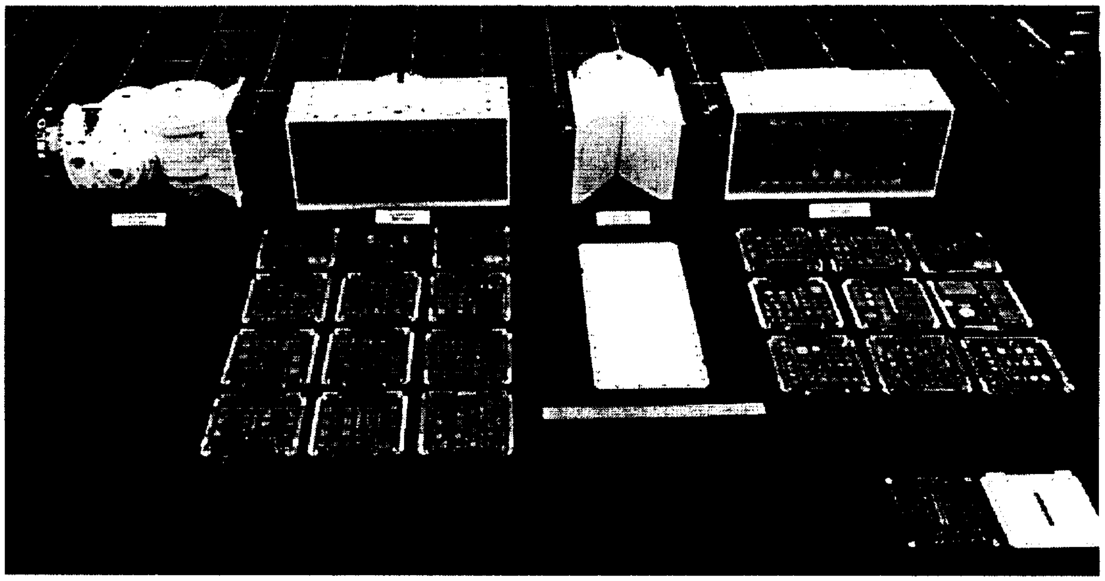


Figure 4 - FTS Manipulator Links and Controller Cards

of cabling, thus minimizing chafing opportunity. The innovative cabling within these actuators is of Flat Conductor Cables (FCC), manufactured by Tayco, Inc. FCC is used in space applications, but for this application up to 34 layers of laminated cables are used in a single actuator passageway. The cables consist of alternating layers of Kapton, FEP, and photo etched copper conductors with a vapor-deposited copper shield. These cables are to operate from -50°C to 95°C through thousands of cycles. These cables route serial data, video signals, power, and discrete signals. Acceptance tests of a few cables indicated minor lamination problems apparently due to entrapped water vapor. Investigation of the cable manufacture and tests of additional cables indicated several areas for possible change as well as a method for cable repair. Recent cable tests to 100,000 mechanical cycles over full temperature ranges verified continued cable functionality.

Control Systems

The FTS manipulator control design provides 6-DOF active control over a wide range of payloads as well as impedance control for stable contact. The wide payload range specified for the FTS manipulator causes the manipulator joints to experience inertial loads over several orders of magnitude. These loads are induced by the coupling which occurs between joints and affects the trajectory-tracking accuracy of the manipulator. The position controller implemented in the FTS manipulator compensates for these torques with a model-based inertia decoupler. The feed-forward decoupling scheme computes expected inertial torques due to commanded motion and sums this torque with the joint command. The position-dependent inertia matrices used to calculate these torques are computed every 200 ms, a time chosen as a compromise of accuracy and computational burden.

In addition to the free-space performance requirements, satisfied with the position controller and inertial decoupler, the FTS manipulator must provide stable contact with its impedance control (see Figure 5). The impedance controller is position-based, that is, the manipulator and joints are treated as actuators of Cartesian position. Thus, end-effector force measurements are transformed into Cartesian motion commands based on a desired output impedance. To maintain stability during the transition from free-space motion to contact, a joint velocity feedback term is included for "augmented damping." The resulted lightly damped contact insures stability, but when contact is broken the free-space motion becomes overdamped and sluggish. A feed-forward velocity term is implemented to compensate for this poor free-space response. These control schemes, which increase the complexity of the controller are designed to meet the FTS free-space motion, payload capacity, and contact performance requirements.

Emergency Shutdown

An emergency shutdown (ESD) system is embedded in the manipulator control architecture. This system was implemented to provide active control of hazards to meet the payload safety requirement to be two-fault tolerant against catastrophic hazards. The primary hazards in this case are unplanned contact and excessive force generation. The ESD approach is to use 3 control levels to monitor joint and Cartesian positions and velocities, comparing both commands and sensor feedback. A separate ESD bus, which connects the joint, manipulator, and power controllers, is the path by which an ESD is initiated - removing power from the manipulator systems. The first level checks that commanded values are within allowable limits both in the manipulator controller and the joint controllers. The second level monitors safety critical parameters such as position, velocity, and torque with the joint controllers and within the manipulator controller collision avoidance routines. The final level of ESD monitoring is a check of redundant safety critical

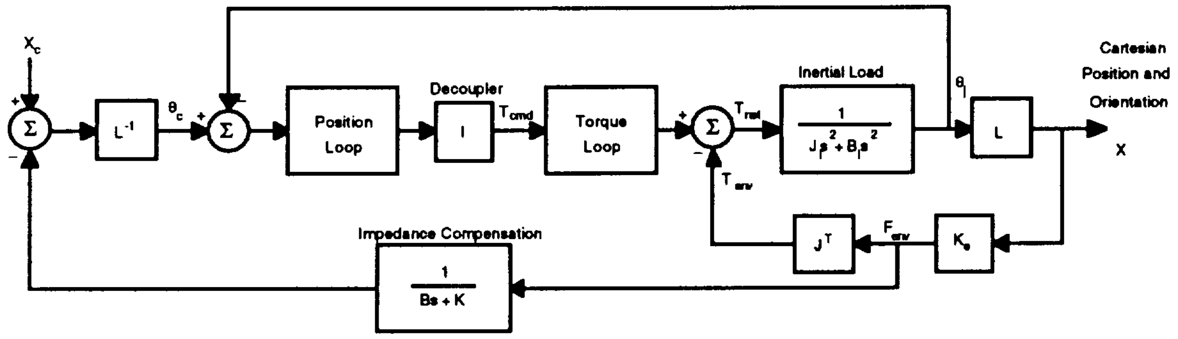


Figure 5 - Manipulator Impedance Control Block Diagram

parameters in the redundant manipulator controller and in independent joint controllers.

In the event of an apparent failure, several possible ESD actions may be automatically initiated. The operator, of course, has a manual ESD to power off the manipulator at any time. If monitored values are elevated but do not pose immediate danger, a soft stop is initiated by the control software. A soft stop commands the manipulator to hold the current position with brakes off (disengaged). An example of a soft stop condition is a Cartesian manipulator command which violates a warning boundary near a known obstacle. A hardware ESD is initiated by any controller when an analog sensor value exceeds its limit value - resulting in an ESD notification on the ESD bus. These analog comparisons are being performed at 1500 Hz. A software ESD occurs when a controller CPU detects an out-of-limit condition and signals the power module over the Mil-Std-1553B communication bus. The power module then initiates a combination ESD to power off the manipulator. A combination ESD is detected by software comparisons in the controllers and initiates a software reset of a hardware limit value to force a hardware ESD. All these ESD paths were analyzed to determine reaction times to various failures such as a joint runaway. Hardware ESDs occur in 11 msec, combination ESDs occur in 30 to 206 msec, and a combination ESD may take up to 4026 msec for an over-temperature condition.

Gripper/End-of-Arm Tooling

The end-of-arm tooling built for the FTS manipulator has a parallel jaw gripper and space for later addition of an end-effector exchange mechanism. This gripper and wrist mounted camera and lights are shown in Figure 6. The gripper fingers are a cruciform designed for positive contact and retention. The gripper fingers ride on a rack and pinion driven by a harmonic drive transmission and a single DC-motor. A pair of fail-safe brakes are installed to provide fault tolerance against inadvertent release. Each of the two brakes can withstand forces greater than expected gripper forces (maximum anticipated load is 30 lb, brake hold is 50 lb). Gripper forces are measured by a torque sensor and also by motor currents. The concern over inadvertent release also impacted the design of planned task items. These items were instrumented to insure positive grasp. As a final safety measure, the gripper fingers are attached with EVA compatible bolts which may be removed on-orbit to release the gripper.

Safety Requirements

Robotic Manipulator Systems can provide the capability to perform work and assist humans in space as long as they are safe and reliable. The space based requirements differ significantly from

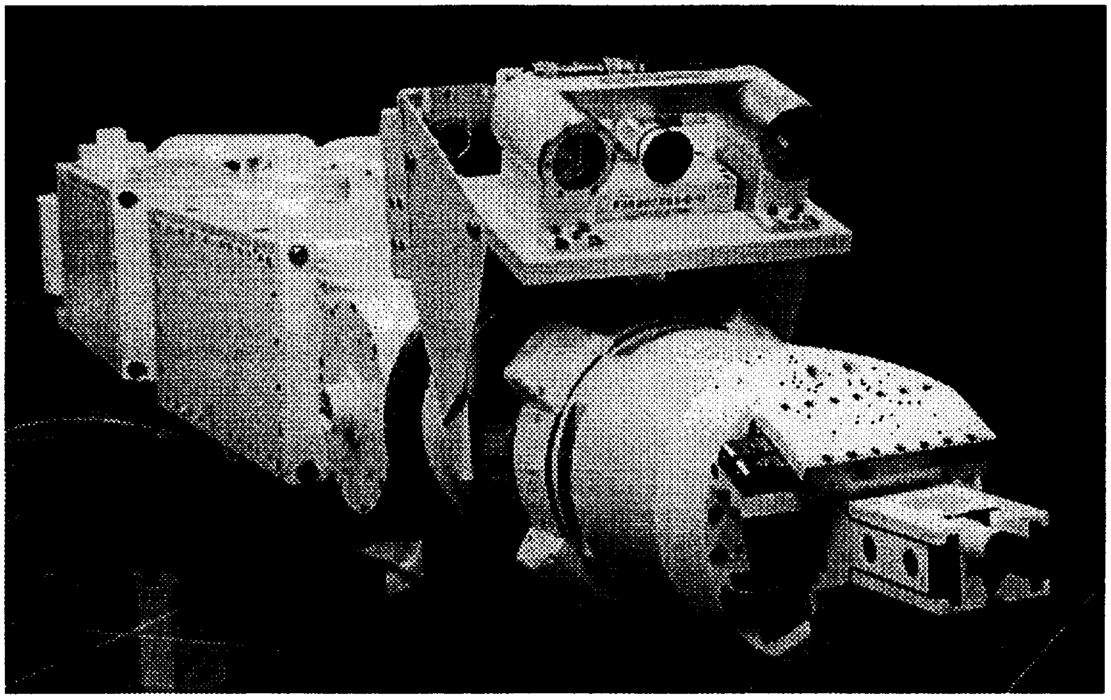


Figure 6 - End-of-Arm Tooling/Gripper

terrestrial based manipulators used in industry and research. In most terrestrial robot implementations, the prime method for dealing with failures is to keep workers out of the robot workspace when active and by accepting the occasional parts damage following a failure due to high volume parts fabrication. This approach is not acceptable for space applications where humans are involved, the effects are very high in costs or it's extreme difficulty to repair. These effects impact the design requirements for space manipulator systems.

Hazards and Controls

All manned space flight systems are assessed for flight hazards their use would impose. From such an assessment the causes of those hazards are determined, and methods to control those hazards are developed. To gain flight acceptance, multiple levels of hazard control must be designed and verified to assure the desired level and coverage of controls. In the FTS system development, safe control of hazardous operations forced additional requirements in the design of the manipulator system, its interfaces with the Orbiter and the task elements the FTS was to interact with.

The primary hazards associated with the FTS manipulator operations and the three methods for providing safe control are as listed:

A) Unplanned contact or impact during operations
   1) Operator and computer control to not command unplanned contact.
   2) Boundary management software operation.
   3) Redundant boundary management software operation in the safety computer
B) Inadvertent release of hardware
   1) Hardwired enable gripper brake power from independent switch in the aft flight deck
   2) Operator Interface Computer: (the aft flight deck portable laptop computer) command to release gripper Brake #1
   3) Hand controller switch to release gripper Brake #2
C) Failure to stow for safe Orbiter landing
   1) Normal computer operations (With hardwired control for added reliability)
   2) Jettison via RMS (or EVA if time permits)
   3) EVA operations to stow or jettison
D) Excessive applied gripper force or torque
   1) Force control using gripper force sensor
   2) Current limiting ESD (Emergency shutdown detection)
   3) Redundant current limiting ESD
E) Excessive applied manipulator force or torque
   1) Normal control with active Cartesian load from joint torque command
   2) Cartesian force limiting, using wrist force/torque sensor channel A
   3) Redundant Cartesian force limiting, using wrist force/torque sensor channel B.

Mission Operation To Control Hazards

Primary concerns in the design of space manipulator systems have to do with the effects of system failures on the crew or vehicle. Operational limitations of use are placed on robotic systems that may otherwise be perfectly capable of performing their intended operations. Limitations on use are imposed due to the fact that if a system is performing a task and were to have a failure, the effect of that failure must not prohibit the intended function from being performed in the time frame that function is critically needed, and any failure must not prohibit any other safety related operations from being carried out during its time of criticality.

For a system to continue operations after a failure, any remaining operability the system might contain must also provide that same capability to make itself safe to the vehicle and crew if it were to suffer a failure. Otherwise that additional level of operability would only be allowed for temporary use to make the task situation safe, remove the robot from the task area, and then stow it in a safe returnable state or eject it so the vehicle can return to Earth. The added operability would not be allowed for continued use to proceed with the intended task, except to make the situation safe. This is the fundamental concept of hazard control for the Orbiter.

FTS Fail Safe Operations

Several FTS configuration descriptions follow below along with design features to address key functions which allow for safe operations. The designs comply with NASA's Orbiter safety policy and requirements of NSTS 1700.7B with interpreted in NSTS 18798A. In several cases, the hardware or software system could not be designed to meet the required levels of fault tolerance without significantly complicating the design or dexterity of the manipulator system. Therefore reductions in compliance with the safety requirements placed operational limitations on the use of the FTS System. The system is considered fail safe; where under any failure the system will not cause a catastrophic hazard, and therefore does not jeopardize the safety of the Orbiter or crew. The FTS system is not fail-operational. Such a system, after any initial failure, could continue normal intended operations since it would still retain the ability to make itself safe after a second failure.

The DTF-1 concept fulfills the first method of hazard control for Orbiter safety using its normal modes of operation. If any of the single points of failure occur, normal operations will cease and an attempt to safe the manipulator system by use of the hardwired control. Note that hardwired control is only a supplement to the first level of hazard control. If the manipulator system cannot be safed by use of the hardwire control, the mission will be assessed to determine if enough time remains to perform an EVA to safe the manipulator system. If hardwired control cannot safe the manipulator system and time does not permit an EVA to safe the manipulator or remove it for stowage, then the RMS will grapple the telerobot using the RMS grapple fixture for jettison. This is the second method for hazard control. The third method of hazard control to provide two fault tolerance for Orbiter safety is EVA operations. Remedial operations could be to remove the manipulator, release the gripper and/or release the actuator brakes. This would be to allow stowage of the manipulator, either into its caging devices or by removal and strapping it in the airlock, or otherwise by release into orbit.

Hardwired Control

The FTS system incorporates a backup hardwired control capability in the event of a failure which precludes closed loop computer control of the manipulator system. The main purpose is to minimize the likelihood of having to jettison the system or perform an EVA operation. This has the effect of making the computer system, sensor systems, software, servo systems and most other hardware single fault tolerant, even though the operations would be significantly degraded in performance.

Operational use of the hardwired control is limited to safing of the system after a failure, by stowing the arm to allow a safe Orbiter return. It allows operator control of individual manipulator joints for stowage and for gripper actuation in the event of computer control or motor drive failure. When selected, primary power is removed from all manipulator motor and brake drivers while retaining power to camera controls. Software recognizes the status of the hardwire control, and commands off all motors and brakes, so that return to normal computer operations after hardwired control starts with all motors and brakes powered off.

Hardwire control is limited to very low joint rates and torques. Hardwired control is by sequential, joint-by-joint movement, and provides no force accommodation to minimize forces imparted into interfaces. Only a limited set of initiated tasks are likely to be able to be completed. Emergency shutdown detection (ESD) is not operational during hardwired control operation, as the operator can de-power the hardwired drive to stop payload motion, and brakes can also be used to stop motion.

EVA Operations

Several failures of components employ EVA as the third hazard control path to ensure stowage of DTF-1 for safe return of the Orbiter. The manipulator actuators, gripper mechanism, and manipulator caging mechanisms represent major groups of such components.

Failure of a caging mechanism to release the arm for operation would not require EVA for safing the manipulator. EVA would be used as the third path for safing the manipulator if more than one of the four caging mechanism fail to close. In this case, removal of the manipulator at its shoulder interface and either manual release into orbit or stowage in the airlock would be required.

Failure of a manipulator actuator motor drive electrically or mechanically would require EVA as the third controlled path. Mechanical release of the joint actuator brake allows EVA backdrive of the joint into the caging position. If a manipulator joint seizes, then EVA is employed as the third hazard control path to remove the manipulator at the shoulder and release into orbit or stowage in the airlock.

Single-Points Failures:

There are several single point failures that remain in the FTS system which may lead to failure of the manipulator to complete a task, or to stow itself for a safe Orbiter return. For the Orbiter this is considered a catastrophic hazard, therefore the requirements for payloads to provide two fault tolerant methods of dealing with these effects.

The FTS single-point failures which lead to an EVA or jettison are few in function, but have commonalty within the actuator and gripper. These failures are seized bearings or gears, a short within the motor winding, or a short or open in a brake winding.

Safety Critical Subsystems

The DTF-1 Flight Experiment of FTS has fifteen different safety critical subsystems and equipment groups, as listed:

Structure Subsystem, Manipulator, Controls, Data Management and Processing, Vision, Sensors, Software, End-of-Arm Tooling, Electrical, Power, Electromechanical Devices, Thermal Control, Task Panel Elements, Aft Deck Workstation, and Hand Controllers

Current Status

The flight FTS manipulator assembly and initial tests were completed under the FTS Technology Capture program at Martin Marietta Astronautics, Denver, in July 1993. An acceptance test and demonstration occurred July 28, 1993, with NASA participation by JSC and LaRC. The tests were conducted on an air-bearing table with all seven joints active, but only four commanded to move for joint and coordinated Cartesian control. The joint servo controller loops had not been individual tuned, and therefore this testing constituted only a demonstration of operation, rather than a performance test. Contact stability and variable compliance interactions with external structures were also demonstrated. The servo tuning can be readily accomodated, as all parameters are programmable, including the torque loop frequency responses.

A follow-on effort called the Bridge Task integrated and checked-out the flight End-of-Arm Tooling (EOAT or gripper) and wrist camera onto the flight manipulator. The Mil-Std-1553B communications bus underwent performance tests between the three internal arm control computers and external coordinating controllers. Martin Marietta provided engineering assessments for a proposed flight experiment concept that separated the manipulator arm from the main avionics. The integrated safety design and control of the system was meticulously maintained. All engineering, analysis, data files and article data packages are being completed and documented under the guidance of Martin Marietta's QA and NASA's SR&QA to maintain the flight heritage of the manipulator and components.

NASA Flight Plans

JSC developed an Orbiter based flight experiment concept to demonstrate a dexterous robotic manipulator system that can operate on the end of the Shuttle Remote Manipulator System (SRMS). This configuration was recommended by Shuttle payload and operations managers as the most useful and beneficial, as opposed to a relocateable dexterous device only. The operational uses allow planned payload manipulation tasks and provides a capability for contingency operations for payloads and for some Orbiter problems. The benefit is to minimize overall EVA time currently consumed by routine tasks, such as EVA site setup and takedown. This would allow EVA to be most usefully allocated for complex operations. Langley Research Center and JPL are team participants in this proposed venture, called DOSS for Dexterous Orbiter Servicing System. Langley would be responsible for advanced robotic controls development and JPL for advanced operator control from a ground control station.

The other significant function of DOSS includes ground control of the dexterous manipulator using 3-D graphic simulations in predictive displays to compensate for the time delays. Ground control allows multiple rotations of ground controllers to operate the dexterous manipulator. The flight experiment concept is cost effective, in that the most expensive development item, the flight manipulator, is available and can be capitalized on. The manipulator along with all ancillary avionics and mechanisms were designed to meet the integrated and operational Orbiter payload safety requirements. Such a flight experiment would provide significant risk mitigation for robotic applications in space, e.g. the new space station, since much of its maintenance is now baselined with the use of ground controlled robotics. The station program seems to be counting on dexterous robotics with no flight operations time to provide insight into possible complications.

REFERENCES

Shattuck, P. L., and Lowrie, J. W., Flight Telerobotic Servicer Legacy, SPIE Vol. 1829 Cooperative Intelligent Robotics in Space III, 1992, pg. 60 - 74.

Andary, J. F., Hewitt, D. R., and Hinkal, S. W., The Flight Telerobotic Servicer Tinman Concept: System Design Drivers and Task Analysis, Proceedings of the NASA Conference on Space Telerobotics, Vol. III, January 31, 1989, pg. 447 - 471.

Flight Telerobotic Servicer Development Test Flight (DTF-1) Final Report, MD-15, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, 1991.

Space Station Flight Telerobotic Servicer (FTS) Phase C/D DTF-1 Phase II Safety Compliance Data Package for Payload Design and Flight Operations, 01-PA-32-08, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, August 1, 1991.

Space Station Flight Telerobotic Servicer (FTS) DTF-1 System Critical Design Review, 01-MD-02-CDR-02, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, October 2, 1990.

Space Station Flight Telerobotic Servicer (FTS) Design Criteria Document, 87600000005 Rev. N, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, March 23, 1990.

Space Station Flight Telerobotic Servicer (FTS) DTF-1 Manipulator Subsystem Specification - Final, 01-TR-20-MS-06, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, December 4, 1991.

Space Station Flight Telerobotic Servicer (FTS) DTF-1 Control Stability Analysis and Simulation Report, 01-TR-05-02, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, October 11, 1990.

Space Station Flight Telerobotic Servicer (FTS) DTF-1 Control Stability Analysis and Simulation Report - Preliminary, 01-TR-05-01, Martin Marietta Astronautics Group, submitted to NASA Goddard Space Flight Center under contract NAS5-30689, July 10, 1989.

4039

N94-30543

AIAA-94-1192-CP

A SPACE STATION ROBOT WALKER AND ITS SHARED CONTROL SOFTWARE

517-63
[illegible]
P. 8

Yangsheng Xu, Ben Brown

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Shigeru Aoki, Tetsuji Yoshida

Space Project Office
Shimizu Corporation
Tokyo, Japan

Abstract

In this paper, we first briefly overview the update of the Self-Mobile Space Manipulator (SM^2) configuration and testbed. The new robot is capable of projecting cameras anywhere interior or exterior of SSF, and will be an ideal tool for inspecting connectors, structures, and other facilities on SSF. Experiments have been performed under two gravity compensation systems and a full-scale model of a segment of the Space Station Freedom (SSF). This paper then presents a real-time shared control architecture that enables the robot to coordinate autonomous locomotion and teleoperation input for reliable walking on SSF. Autonomous locomotion can be executed based on a CAD model and off-line trajectory planning, or can be guided by a vision system with neural network identification. Teleoperation control can be specified by a real-time graphical interface and a free-flying hand controller. SM^2 will be a valuable assistant for astronauts in inspection and other EVA missions.

1 INTRODUCTION

Since 1989, we have been developing the Self Mobile Space Manipulator (SM^2) which is a walking robot to assist astronauts on the Space Station Freedom and other space structures in performing construction, maintenance and inspection tasks. It has end-effectors for attachment, and can step from point to point to move freely around the exterior of space structures. SM^2 can replace EVA astronauts in performing tedious or dangerous tasks, and can be deployed quickly to investigate emergency situations. It is simple and modular in construction to maximize reliability, simplify repairs and minimize development time. SM^2 is lightweight, so it can operate with minimum energy and disturbance to the structures.

Over the past four years, SM^2 has progressed from concept, through hardware design and construction, to software development and experiments with several versions of the robot. During the first year, we developed a concept for robot mobility on the space station trusswork, and experimentally tested a variety of control algorithms for simple one-, two- and three-joint robots. During the second year, we developed a simple, five-joint robot that walked on the tubular-strut-and-node structure of the original Space Station Freedom design, and a *gravity compensation* system that allowed realistic testing in a simulated zero-gravity environment. The third-year work focused on development of the manipulation function; we added a part-gripper and extra joint at each end of the robot, and developed related control software.

In this paper, we will report the research and development work performed during the forth year of the project, with emphasis on the shared control system developed to facilitate the execution of complex tasks in space applications.

2 NEW SM^2 DEVELOPMENT

In response to the changing design and needs of SSF, our focus has shifted to adapting SM^2 as a mobile inspection robot to augment the fixed video cameras planned for SSF. The robot's size and configuration have been adjusted to accommodate the new truss structure. The space station truss design has been changed by NASA in favor of the current pre-integrated truss (PIT) design, utilizing I-beam members. The new truss design is hexagonal, rather than rectangular in shape. Therefore, our first goal was to

[0]Copyright c American Institute of Aeronautics and Astronautics, Inc., 1994. All rights reserved

modify the SM^2 configuration to adapt to this new space station truss.

The second goal of the project was to specialize the SM^2 robot as an inspection robot. There is a vital need for inspection of facilities on the space station, such as fluid connectors, electric cables, and bolted segments. Able to reach both exterior and interior of the space station, the movable cameras will be essential for this task. SM^2 will be capable of projecting cameras to any position on the space station through its inherent self-mobility.

2.1 Robot Configuration

The robot's size and configuration have been adjusted for the new truss structure as shown in Figure 1. On the previous truss design, five degrees of freedom (DOF's) were sufficient for locomotion from any given node to any adjacent node. The robot had two joints at each tip and one elbow joint. In order to enable the new robot to step from one face of the redesigned hexagonal PIT structure to adjacent faces, and to retain the symmetry of the SM^2, the new robot requires a total of seven joints, three at each tip and one at the elbow. The symmetry of the robot mechanism is important for the control of locomotion, so that as the base of the robot is switched, we simply switch the numbering of the joints from the base to the tip. This allows the out-of-plane motion needed to step from one face of the truss to another. In addition, the total length of the robot has been increased, and the flexibility of the two long links has been reduced so as to accommodate the size of the new truss design, while still maintaining the low mass essential for space applications.

Each of the seven joints is identical, self-contained and modular so that a minimum inventory of parts is required for joint repair or replacement. The joints are driven by harmonic motors and are wired in a modular fashion so that only one 16-pin connector is required to deliver all signals and power to each of the joints.

2.2 Beam Grippers

The new truss structure made the old node grippers obsolete and required design of new grippers that could attach to the aluminum I-beams of the PIT structure. Each end of the SM^2 is now equipped with a three-fingered gripper capable of grasping I-beam flanges of various thickness and width, as shown in Figure 2. The single finger, driven by a DC motor, slides back and forth to allow opening and closing of the gripper. A linear potentiometer measures the single finger position, while motor current indicates grasp force.

Each gripper has been equipped with sensors necessary for reliably and securely grasping the beam. Using force-sensing resistors, contact switches on each of the three fingers can be checked to verify a good grasp. In addition, capacitive proximity sensors at the base of the fingers sense beam proximity up to about four inches away and are useful in aligning the gripper with the beam.

2.3 Cameras Modules

There are three camera modules attached to the robot, one at each tip, and one on the elbow joint. Each camera has separate controllable zoom, focus, and iris with four high-intensity lamps arranged around each camera.

The elbow camera has one motorized degree of freedom. Since the robot has one redundant DOF, the elbow camera has effectively two DOFs in determining it's view. With both ends of the robot attached to the truss, for example, the collection of all possible views sweeps out a half torus about an axis defined by the two base joints at each tip. Thus, the elbow camera can provide valuable visual information about global location on the space station.

The two tip cameras serve twin purposes. The primary purpose is, of course, visual inspection by human operators. The robot tip camera at the free end can provide views of the truss structure that any fixed camera around the space station simply cannot achieve. I-beam connections as well as the inside faces of the I-beams are two locations where a movable camera might provide significantly better views. The secondary purpose for the cameras concerns autonomous locomotion on the truss. We use neural-network based machine vision with images from the tip camera to autonomously mate the gripper to the I-beam flanges. The tip camera module and end-effector are shown in Figure 3.

2.4 Gravity Compensation

To simulate the zero gravity environment of space, we use two independent gravity compensation systems developed at Carnegie Mellon University. Each gravity compensation system provides a constant upward vertical force through a counterweight mechanism and a series of cable and pulleys. The support cables are attached to the centers of gravity of the two long links on the robot. A 10:1 ratio in the counterweight mechanism keeps the increased inertia in the vertical direction to 10

The support cables attached to the robot are tracked overhead by two separate, actively controlled carriage systems. Angle sensors detect x-y movement of the support cables. The first system is a Cartesian gantry system and allows robot motion in an area that is 17 feet long and 9 feet wide. This allows us to test large global stepping motions for the robot. The second system is a smaller cylindrical compensation system supporting a smaller field of motion. This allows a large variety of motions to be tested without the supporting cable of the larger system interfering with the carriage beam of the smaller system [2].

In addition to the mechanical gravity compensation, we provide for active residual gravity compensation in software to correct for minor discrepancies in the mechanical system. This is especially necessary to provide appropriate torques for the three joints at the free end of the robot. The combination of mechanical and active gravity compensation provides for realistic zero gravity experiments and testing.

2.5 Truss Mock-up

In our lab, we have built a truss mock-up which is a full-scale representation of a small portion of the entire truss structure on the space station. The mock-up includes four faces of the hexagonal structure as shown in Figure 4. Each beam is constructed of wood with sheet aluminum laminated to the flange faces to allow for realistic machine vision testing. Varying flange widths and thicknesses allow for robust testing of the grippers.

3 REAL-TIME SHARED CONTROL ARCHITECTURE

At the heart of the SM^2 control software lies a real-time shared control architecture [1]. It is modular in design whereby tasks are composed of independent, reusable subtasks. High level tasks for the SM^2 robot range from teleoperation to semi- autonomous tasks to fully autonomous walking. These tasks often use many of the same subtasks such as trajectory tracking, beam grasping, point convergence, and switching the base of the robot. These subtasks are coded as modular library routines which may be dynamically sequenced through a coordination module and state machine.

3.1 Coordination of Tasks

The various task modules need to be coordinated in an intelligent fashion. We used a state machine, programmable through a simple language and parsed in real-time. The state file describes the following attributes of the state machine:

- Defines the number of subtasks and the possible message inputs and outputs for each subtask.
- Defines all tasks (states).
- Defines all possible transitions and the initial task (state).

A subtask is defined as shown in the following example:

```
SUBTASK   grasp
INPUT     on off open close stop gripper1
          gripper2
OUTPUT    noncontact contact done grabbed
```

The first line merely assigns a label to the subtask. The second line gives a list of valid messages that the subtask *grasp* will accept as input. Each of these inputs is easily understood. For example *open* commands the subtask to open the gripper, while *gripper2* commands the subtask to switch to gripper2. Finally, the last line specifies the outputs of the subtask.These are then used in the sequencing of states.

A typical task specification might appear as follows:

```
TASK       tele_gripper_close
SUBTASKS   grasp tele
START      tele:on tele:grp grasp:close
END        grasp:off
```

Here, again, the first line merely assigns a label to the task. The second line specifies which subtasks are part of the overall task. In this example, both *grasp* and *teleoperation* combine to form the specific task. The next line specifies what messages to send to the various subtasks at the start of the overall task. The first two commands make certain that teleoperation is in the *on* mode and that the control mode is the *gripper mode.* The final start message instructs the *grasp* subtask to attempt to close the gripper. In the final line, we specify what messages to send at the end of a task execution. Once the gripper is closed, we instruct the subtask *grasp* to turn off.

Finally, below we show an example of specifying state transitions and an initial state:

```
TRANSITION     tele:down tele_gripper_idle
               tele_gripper_close
INITIAL_TASK   tele_init
```

The transition statement simply states that when the subtask *tele* receives a *down* message - when the appropriate button is pressed on the teleoperation hand controller - the state machine should sequence from the *idle gripper* mode to the *close gripper* mode.

In such a manner, high-level tasks can quickly be programmed from a library of subtasks through the state machine. Note that subtasks are reusable from state to state and can be switched on and off when necessary. For example, the *grasp* subtask is equally necessary in the autonomous locomotion mode as well as the teleoperation mode.

In short, the state machine allows subtasks to be shared by high-level tasks which can be rapidly re-programmed with minimal re-coding and no re-compilation. This allows for elegant and rapid software development.

3.2 Task Modules

We have developed several reusable task modules for the SM^2 control software. In each control cycle, the task modules perform four basic functions:

- Read messages from the state machine and respond in appropriate fashion.
- Read sensor devices, global variables, or receive input from remote tasks.
- Generate desirable control motion based on local inputs.
- Send appropriate messages to the state machine.

Since each subtask module produces desired control commands based solely on its limited criteria, one module - the *combination* module - is required to intelligently combine these desired control outputs from individual task modules into one coherent control signal. The combination module therefore ensures reasonable control outputs based on a weighted average of the control commands of the individual task modules.

Remote task modules do not fundamentally differ from other modules except in one respect. These modules are run on a separate workstation or processing board, usually due to high computational requirements that cannot be met in real time. These modules can interface with the slower real-time boards via UNIX sockets, a VME bus, or serial lines. Menu-driven user interfaces as well as a real-time graphical displays are two examples of such computationally intensive remote tasks. These, along with the other task modules will be discussed in the context of the following two sections which discuss (1) autonomous walking on the truss, (2) and teleoperation.

4 AUTONOMOUS LOCOMOTION

4.1 Model-Based Walking

The operating environment for the SM^2 is very structured and can easily be modelled with a great degree of accuracy. Hence, it is possible for the robot to execute a pre-planned sequence of walking steps based solely on a model of the space station truss structure. We have successfully executed various sequences of four steps on the truss mock-up, including steps of variable length and between different faces of the hexagonal space station truss structure. Each walking step is decomposed into several distinct phases: (1) ungrasping the beam, (2) separating smoothly from the beam, (3) executing a global trajectory, (4) executing a straight-line motion towards the beam, (5) closing the gripper, and (6) switching the base for the next step.

First, the gripper is opened until the sliding potentiometer indicates that the gripper is in the fully opened position. Second, while keeping the orientation of the gripper aligned with the beam, the free end is moved above the beam in a straight-line motion so as to avoid potential collisions with the space station truss. Once the free end is safely above the truss structure, control is switched to the execution of a global trajectory in the state machine.

A global trajectory is defined minimally by the starting point and the target destination. The operator, however, is free to include as many via points as he chooses along the path of the trajectory. These points may be generated alternatively in a preprogrammed file or through the real-time graphical display as discussed in the subsequent section. As the trajectory is being executed, errors are dynamically corrected by continuously calculating a smooth path between the current position and the desired trajectory path. If no, intermediate points are specified along the trajectory, the inverse kinematic algorithm, as explained later on, will generate intermediate points which lead to a smooth trajectory.

The trajectory will finish with the proper gripper orientation about 20 inches above the target beam and location. From there, the state machine enters the next phase of execution; that is, a straight line descent towards the target beam along the surface normal of the beam.

Each gripper has multiple sensors that can be used during approach to the beam and grasping. Proximity sensors at the base of each finger provide information about the relative orientation of the gripper and beam from several inches away, and signal when the gripper face is close against the beam. Contact switches, using force sensing resistors (Interlink), sense contact of the three fingers with the edge of the beam to verify a sense grasp. Gripper motor current is also sensed to indicate the grasp force. After the initial grasp is made, the gripper is opened slightly (about 0.25 inch) and closed again. This helps to automatically correct for any remaining misalignment.

Finally, if another step is to follow, the robot will switch bases. What was the free end before, will now become the fixed base and vice versa.

It is important to note that the entire sequence described above is controlled through the state machine. Each phase of the stepping motion will execute only when the appropriate *done* message is sent by the control software to the state machine. The proper *done* message triggers a transition to the next state. The entire walking step is divided into a sufficient number of subtasks, any or all of which can be used during other modes, such as teleoperated or semi-autonomous control.

4.2 Neural Network Based Visual Servoing

Although we have a good model of the environment, errors can accumulate over consecutive steps. This can potentially lead to a failure in properly grasping the next beam. If this should occur, a neural-network based vision system will assume control, correct any such error and properly complete the grasping of the beam. It is preferable to use the vision system only when failing to complete a grasp, since the vision system slows the system performance significantly. The main bottleneck is, of course, the acquisition of the images at a high rate.

We trained a neural network on 40x40 digitized images of flanges at various translational offsets, heights, and rotations. The neural network learned through the standard back propagation learning algorithm.

Once the vision system has placed the gripper in contact with the beam, the state machine returns control to the same states and subtasks used for closing the gripper as mentioned previously.

Unlike the previous strut-and-node design of the space station truss structure, the current design causes uncertainty in the location of the robot on the truss structure, since SM^2 is free to grasp the beam anywhere along its length. That uncertainty could potentially be periodically removed by using the vision system to locate certain known special locations on the space station truss. One such special feature might be where two or more beams join. Further work needs to be done in this direction.

5 TELEOPERATION

We have developed two different methods for teleoperation. The first method utilizes a six-DOF hand controller to guide the free end of the robot. The second method utilizes the real-time graphics display which provides two views of the space station truss structure. By selecting the target location for the robot arm with a mouse, the robot can be made to execute large global trajectories.

5.1 Hand Controller

We use a commercial, six-DOF, free-flying hand controller as the principal means for teleoperated control. The device, called the *Bird*, operates with a stationary radio transmitter and a moving receiver. Both the position and orientation of the receiver relative to the transmitter is communicated via a serial line to the controller at a rate of 10Hz. The moving receiver is attached to a cylindrical stick with an enable switch controlled with the thumb, and another multi-purpose two-way switch controlled with the index finger. Figure 5 shows the control station configuration and the use of the hand controller.

The hand-controller is used in conjunction with a graphical user interface to determined the mode of operation for the hand controller as well as the function of the two-way switch. The menu-driven user interface allows the operator to select one of three basic modes of operation, as well as which end of the robot is the active one. The three modes are (1) position control, (2) velocity control, and (3) gripper control.

In gripper mode, the two-way switch controls the opening and closing of the gripper. Velocity control is generally used during large global motions of the robot, while position and gripper control are used when grasping a beam and switching the fixed base of the robot.

In each mode, the operator can select whether the motion of the free end of the robot is to be base-relative, tip-relative, or semi-autonomous. Tip-relative motion is generally the most useful when the only visual feedback for the operator is from the elbow and tip camera (i.e. the robot itself is hidden from view). Base-relative motion is useful in conjunction with either fixed camera views or the real-time graphical display which reveal the global position of SM^2 on the space station truss.

In manually mating the free end of the robot to one of the I-beam flanges, the semi-autonomous mode simplifies the process for the operator. The semi-autonomous mode allows the operator to automatically orient the free gripper to the correct orientation for grasping the beam. The control software utilizes knowledge of which beam the fixed end is currently attached to and which beam the operator wishes to grasp in order to select the proper orientation for the gripper. With this semi-autonomous orienting, the process of teleoperated walking on the space station truss is significantly facilitated. Requiring only minimal training, we have repeatedly demonstrated teleoperated walking on the truss mock-up, with and without the robot in view of the operator.

The above discussion illustrates several dimensions of the shared control architecture. We achieve a blend of teleoperation and autonomous locomotion without the need for new software code. In the semi-autonomous teleoperated mode, we use the same subtask to achieve the proper orientation of the gripper before grasping as we do in autonomous walking. Furthermore, we are able to use the same grasping subtask for autonomous walking and teleoperation. In fact, the message to the state machine issued during autonomous walking and teleoperated control is exactly the same: *close gripper*. Thus, all the safety precautions used for ensuring a secure grasp of the beam during autonomous walking are automatically incorporated when the operator commands the gripper to close on the beam.

In another example, the operator may wish to inspect the length of a beam. Rather than worry about following a precise straight line with the hand controller, the operator may wish to surrender control of one directional degree of freedom (transverse to the beam) so that he can inspect the length of the beam with variable speed, approaching the beam closer if some damage is observed. This may be achieved by employing the same trajectory subtask as is used for the autonomous walking. Again, the shared control architecture allows an elegant merging of autonomous and teleoperated function. Simply with some minor additions to the state machine, the teleoperation function is seamlessly incorporated into the overall control architecture.

5.2 Real-Time Graphical Interface

Rather than explicitly define the trajectory which the robot is to follow, an operator may wish to simply specify starting and stopping points for global stepping motions. To this end, we have developed a real-time graphical interface.

The graphical user interface is a PHIGS and XView-based application which runs as a remote task module. It has been designed to perform the following functions:

- It provides a 3D display of the robot position, configuration, and its location on the space station truss structure. Ambiguities in the 3D display on the 2D screen are resolved by providing two separate, modifiable views.
- It allows for manually controlling task sequencing in the state machine in real-time.
- It serves as a teleoperation input device for controlling global robot motions.
- It allows for visually pre-planning and simulating robot stepping motions to avoid obstacles and singular or near singular configurations.
- It serves as visual feedback to an operator by providing a global view of the robot on the space station truss. In addition, it warns of potential collisions by sending appropriate messages to the state machine. The operator can thus modify the robot trajectory accordingly.

In teleoperation mode, the graphical display translates mouse commands into trajectories in real-time. Once again, teleoperation and autonomous function are combined through the shared control structure. After the operator specifies desired steps for the robot, the same subtasks which perform autonomous walking are employed.

6 CONCLUSION

The SM^2 robot has been redesigned to be compatible with the new space station truss structure. Both the software and hardware of the SM^2 system has been designed to be modular, in order to shorten repair, maintenance, and development time. We have demonstrated both autonomous walking as well as teleoperation functions in a single shared control architecture. Depending on the calibration errors, the model-based locomotion with off-line trajectory planning, and neural-network based vision can be used for reliable walking. The real-time graphics interface provides a valuable tool for specifying control inputs in teleoperation and for displaying the robot configuration under communication delay. The free-flying hand controller provides an easy way to command robot action with two monitor views from the robot cameras.

Acknowledgements

The project is supported by Shimizu Corperation, Space Project Office. We would like to thank the following colleagues for their technical contribution and support: Alex Douglas, Michael Nechyba, Randy Casciola, Mark Friedman, and Takeo Kanade.

References

[1] A. Douglas and Y. Xu, "Real-Time Shared Control System for Space Telerobotics," *1993 IROS Conference Proceedings*, pp. 2117-2122, 1993.

[2] Y. Xu, et. al., "Teleoperated Mobile Manipulator for Space Station," *1993 JSME International Conference on Advanced Mechatronics Proceedings*, pp. 830-835, 1993.

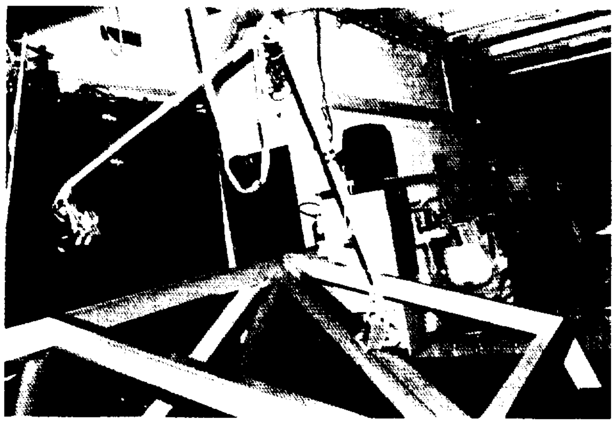


Figure 1: A 7-DOF robot manipulator for space station inspection

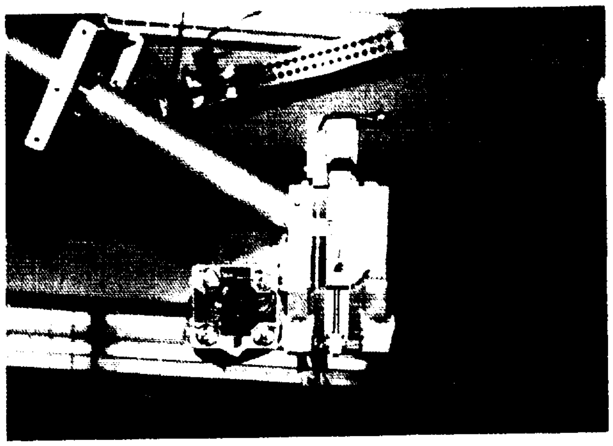


Figure 2: The beam gripper and tip camera module

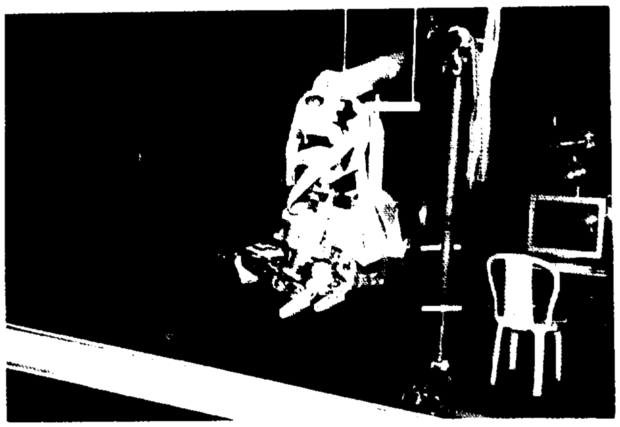


Figure 3: The tip camera module and beam gripper serve as vision guided effector in stepping motion

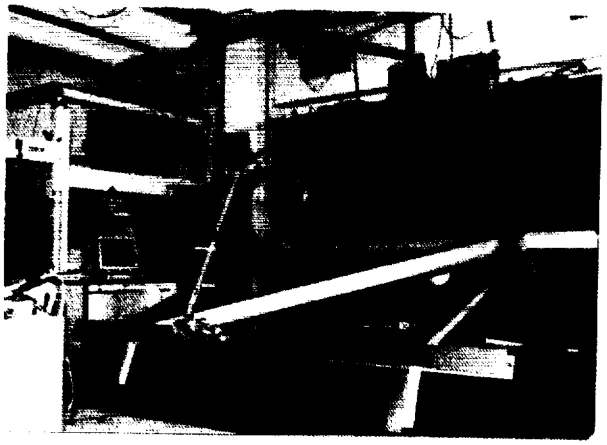


Figure 4: The Space Station Freedom mock-up and telerobotics testbed

Figure 5: The control station with a free-floating hand controller

6034

N94- 30544

AIAA-94-1193-CP

Technology for Robotic Surface Inspection in Space

Richard Volpe[*] **J. Balaram**[†]
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

S3-63
P. 11

Abstract

This paper presents on-going research in robotic inspection of space platforms. Three main areas of investigation are discussed: machine-vision inspection techniques, an integrated sensor end-effector, and an orbital environment laboratory simulation. Machine-vision inspection utilizes automatic comparison of new and reference images to detect on-orbit induced damage such as micro-meteorite impacts. The cameras and lighting used for this inspection are housed in a multi-sensor end-effector, which also contains a suite of sensors for detection of temperature, gas leaks, proximity, and forces. To fully test all of these sensors, a realistic space platform mock-up has been created, complete with visual, temperature, and gas anomalies. Further, changing orbital lighting conditions are effectively mimicked by a robotic solar simulator. In the paper, each of these technology components will be discussed, and experimental results are provided.

1 Introduction

Later this decade, NASA will place in orbit around Earth the Space Station Freedom (SSF), which will be used as a science station and home for astronauts for 30 years. Soon after its initial design, engineering reviews revealed that simple inspection and maintenance of the station would consume more time than the astronauts would have available [2]. This was reinforced by results of the Long Duration Exposure Facility (LDEF), which showed large amounts of damage from micro-meteorite impacts and atomic oxygen degradation while in orbit for five years [8]. For these reasons, NASA sponsored *The Remote Surface Inspection Task* (RSI), a five year technology demonstration task at the Jet Propulsion Laboratory, California Institute of Technology (JPL). This project has developed and systematically investigated methods for telerobotic inspection of SSF [4].

[*]email: volpe@telerobotics.jpl.nasa.gov
[†]email: balaram@telerobotics.jpl.nasa.gov

The inspection system which has been built for this research is comprised of three main subsystems: robot manipulator control, graphical user interfacing, and teleoperated/automated multi-sensor inspection. The robot manipulator subsystem is comprised of a Robotics Research K1207 arm mounted on a translating platform, and controlled by a real-time system employing Configuration Control [9]. The graphical user interface subsystem resides on an SGI workstation and provides user-friendly interfaces to the manipulator control and the inspection data [6]. The multi-sensor inspection subsystem analyzes a realistic SSF mockup under simulated orbital conditions, gathering sensory data indicative of potential problems. This inspection subsystem is the topic of this paper. The key technology items addressed are: methods for automated visual inspection; the development of an Integrated Sensor End-Effector (ISEE) which encompasses vision, proximity, temperature, and gas information to monitor the environment; and a high fidelity simulation of orbital inspection conditions. In this paper, each of these will be described as well as the issues which they successfully address.

The paper is organized as follows: Section 2 discusses automated visual inspection in detail, including the issues of ambient light and registration error compensation, as well as flaw and error models. Section 3 describes the ISEE, and provides a detailed discussion of the use of proximity sensors for collision avoidance and surface following. Section 4 discusses the simulated conditions for the inspection operations, including a description of the SSF truss mock-up and its temperature and gas anomalies, as well as a solar simulator which provides realistic orbital lighting conditions. Finally, Section 5 provides a summary and some conclusions drawn from this technology development research.

2 Visual Inspection

The approach adopted for on-orbit inspection of space platforms consists of locating and characterizing flaw-induced changes between an earlier *reference* image and a new *inspection* image. In the absence of noise, viewpoint differences, lighting variations, and benign changes, the detection of significant new damage could be obtained by a process of simple differencing. However, on-orbit use of robotic machine-vision to achieve this goal is constrained by a number of technical challenges:

- **Imaging Repeatability.** Subsequent scans of the space platform will not be able to achieve the same imaging view-point because of the lack of robot positioning repeatability and the expansion/contraction of space platform structures. This can result in mis-registered *reference* and *inspection* data sets, as well as previously occluded features being made visible and mistaken for new flaws. The presence of the flaw itself can complicate the recognition of the extent of the mis-registration.

- **Lighting Variation.** In orbit, surface appearance can change drastically due to the variation in ambient light (solar and earthlight) illumination induced by orbital motion. Power constraints on artificial illuminators restrict the illumination techniques that can be adopted to compensate for this variability. Furthermore, the lack of atmospheric dispersion of the harsh solar light results in images having a large dynamic range with sharp shadows.

- **Flaw and Object Appearance.** The surface flaws caused by micro-meteorite damage are very small ($\approx$ 1 mm) [7] and must be detected on man-made objects with complex geometric shapes and constructed with specular materials. Benign changes such as the gradual reflectivity variation resulting from exposure to ultra-violet radiation and atomic-oxygen can mislead the inspection system.

- **System Constraints.** Efficient computer processing is a must, given the computational limitations imposed by the need to use compact, light-weight, low-power, space-qualified computers. Communication limitations in sending data back to Earth must also be considered in deciding on the partitioning of the image processing functions between the spacecraft and the ground. Data storage of the various *reference* images is less of a problem than would initially appear, thanks to the availability of space-qualified mass storage devices.

- **Motion Constraints.** Robot motions can induce significant platform disturbances due to robot start/stop motions. If the disturbance is to be minimized by performing all of the imaging from a continuously moving sensor platform, then the resulting problems of motion blur must be addressed.

In this report the focus is mainly on the effects of ambient light variability and image mis-registration, and the methods used to compensate for them. A brief discussion on flaw-models and the quantification of the flaw detection performance is also presented. A detailed presentation may be found in reference [1].

2.1 Laboratory Imaging System

The imager consists of an industrial color Charge Coupled Device (CCD) camera. With solar illumination at earth orbit at approximately 130000 lux, the total illumination on a typical inspection scene area of 0.1 m^2 over the duration of a single video field (1/60 s) is approximately 215 lumen $\cdot$ s. This is many times that which can be provided by a low-powered artificial light source, especially if it were a continuous illuminator. Instead, artificial illumination is provided by an electronic strobe unit, with the laboratory unit providing an illumination of 1.3 lumen $\cdot$ s. When the strobe is used with the electronic shutter in the camera set to 1/10000 s, the total ambient solar illumination of the scene is only 1.5 lumen $\cdot$ s, making it comparable to the strobe provided illumination. Note that the total strobe illumination remains unaffected by the electronic shutter activation because the strobe duration ($\approx$ 20 μs) is still much shorter than the exposure duration. The overall energy consumption for strobe lighting is also lower since the strobe is only used when the sensing platform traverses a new view-point. Further, the use of a short exposure time reduces the effects of motion-blur in degrading the images. (As a practical note, since the laboratory ambient light simulator, described in Section 4, cannot achieve full solar intensity, the camera electronic shutter is operated at a somewhat larger setting. This effectively achieves the same ambient-to-artificial illumination ratio relevant to orbital operations.)

The camera is operated with a unity gamma response. Any deviations from a linear response are compensated for in the digitizer. Linear response ensures that image intensity is proportional to scene radiance and allows linear operations (e.g. subtraction) on image fields to be correctly computed. This is required for the ambient light variability compensation methods discussed in the next section. All imaging is performed using only the luminance signal of the video signal (quantized to 8 bits) with the color subcarrier information suppressed.

2.2 Ambient Light Compensation

Ambient light *subtraction* uses two image data sets to obtain a *compensated* image. The first data set is illuminated only with the ambient light and the second is illuminated with the ambient light as well as the artificial illuminator. The information in the first data set is subtracted from that in the second to give a compensated image that appears as if it were taken with the artificial illuminator alone. In order for the subtraction results to be valid, the sensor response is required to be linear. There is, however, a reduction of the signal-to-noise (S/N) ratio since the subtraction process can nearly double the noise power in the data. Further, the utilization of the dynamic range of the camera is also reduced since the sensor cannot be allowed to saturate when both ambient and artificial illumination is utilized. The performance of ambient light subtraction is enhanced when the artificial illuminator provides energy comparable to (or more than) the ambient light. As discussed earlier, the electronic shuttering mechanism achieves this. Note that strobe illumination is essential here for operating with a moving imaging platform since continuous illuminators, even if low power and high-intensity, would take a finite amount of time to ramp up to the desired intensity level. This would require the imaging platform to be stationary during the taking of the two image data sets necessary to achieve compensation.

An additional problem is that in a strobe illuminated image only one of the 2 : 1 interleaved image fields (say the odd-field) is lit by the strobe, while both fields are ambient lit. An estimate of the ambient light component in the odd-field is generated from an average of the ambient light data in the even-field immediately above and below each odd-field image scan line. A *compensated* image is generated by *intra-frame subtraction*, wherein this ambient light estimate is subtracted from the odd-field data.

This process does suffer from some disadvantages, namely a halving of the vertical resolution in the compensated image and the possibility of interpolation errors when estimating the ambient-lit component of the image. As expected, if the same ambient light is used in the *reference* and *inspection* images, then the interpolation errors are identical and cancel when performing the subsequent image comparisons for flaw detection. Any non-zero change can then be attributed to the presence of a new flaw.

However, interpolation errors are of consequence when the ambient light changes, and lead to an increased probability of false errors during the flaw detection process. For two special cases which correspond to limiting cases typically encountered in real applications, the deleterious effects of the interpolation error is manageable. The first case corresponds to when the ambient light illumination of the surface for both the *reference* and *inspection* images has low spatial variation and the underlying reflectivity of the surface undergoes a large change. Here analysis shows that the significant errors only happen in regions where the reflectivity changes are large, which are precisely the same regions where mis-registration errors due to sensor-to-platform positioning errors can be expected to be of greater significance.

The second case occurs with ambient light discontinuities at shadow boundaries. If the transition from light to dark in the "pen-umbra" region of the shadow is very sharp, then the estimate generated by interpolating the even-field data will be incorrect. If, however, the transition occurs over a spatial extent of more than a couple of pixels, then the interpolation process will be able to accurately estimate the ambient light in the middle of the shadow boundary region. The extent of the pen-umbra region is a function of the distance from the surface to the object casting the shadow. If the object is close to the surface then the transition is sharp, and conversely if it is far away from the surface the transition is more smooth. Assume that a pen-umbra region greater than 2 pixels is of sufficient spatial extent to permit the interpolation to be reasonably accurate. An estimate of the corresponding object distance that would generate such a shadow can be easily obtained from simple geometrical arguments. For a typical field-of-view and imaging standoff-distance, a shadow transition region of 2 pixels corresponds to to a surface spatial extent of about 1 mm. Noting that the sun subtends approximately 0.01 radians, and that the shadow penumbra must necessarily subtend the same angle, gives the corresponding object distance as being 0.1 m. Thus sharp shadows will only be cast by objects closer than 0.1 m to the surface. Even for such sharp shadows, the situation is ameliorated by the fact that the resulting interpolation errors are localized to a region along the shadow boundary that has a very narrow width. If the flaw being detected has a spatial extent larger than this width, then the resulting errors during flaw detection are reduced. This issue is discussed further in the Section on flaw models (Section 2.4).

2.3 Registration Error

Registration errors are induced by the lack of repeatability in the viewpoints at which images are taken for the *reference* and *inspection* images. These viewpoint discrepancies arise due to the inherent accuracy limitations of moving camera platforms. In the laboratory environment, i.e., fixed targets and industrial arms with good repeatability, the inaccuracy translates to no more that one to two pixels when images are taken from relatively short distances of less than 0.7 m. In the space environment, larger repeatability problems are to be expected due to arm flexibility and object location changes due to thermal expansion and structural flexibility.

With this mis-registration, the comparison of compensated images by performing a simple subtraction of the compensated *reference* and *inspection* images results in a number of "false edges" in the differenced image. The magnitude of registration error depends on both the directional gradient of the gray-level image with respect to the camera platform motion parameters, as well as the occlusions at each imaged point. Here, only the directional gradient with respect to lateral and horizontal motion of the camera platform are considered, since these are expected to dominate for this inspection application. Occlusion induced errors are also not considered, even though their effects could be significant near any sharp depth deviations in the image.

A Gauss-Newton iterative method is used to perform *reference*-to-*inspection* image registration prior to making the comparison. The residual sum of squares between the actual and an estimated picture is used as an evaluation function to indicate the degree of match between the *inspection* data and a transformed *reference* image. The objective is to find a suitable transformation of the *reference* image so that the residual is close to zero. The Gauss-Newton algorithm solves this nonlinear least-squares problem via an iterative solution method and exploits the special structure of the gradient and Hessian matrix of the evaluation function [3]. The iteration process is continued until the least-squares residue drops below an acceptable threshold, at which point the estimate can be considered to be registered with the data. Note that terms involving the Jacobian matrix in the case of pure translational mis-registrations can be pre-computed resulting in significant run-time computational savings. Nevertheless, residual mis-registration is still possible because of early termination of iterative registration correction necessitated by real-time deadline processing constraints.

2.4 Flaw and Error Models

The process used to detect a flaw is intimately linked to the corresponding model of the flaw. Flaw models must provide a reasonable approximation to the physical appearance of the flaw while not being overly complex to preclude implementation of the associated flaw detection algorithms on a real-time system. Two types of flaw models are presented and the corresponding flaw detection processes are characterized.

A *single-pixel flaw model* treats each individual pixel independently of other pixels when it comes to flaw determination. A flaw is assumed to be present at a pixel if the surface intensity at that pixel in the *inspection* image differs from the surface intensity in the corresponding *reference* image pixel by a value greater than a characteristic flaw strength. The characteristic flaw strength is a function of the flaw type and can be determined by examining images of known and/or calibrated flaws.

In a *multi-pixel flaw model* a flaw is assumed to be present at a pixel if it occupies a certain minimal spatial extent. More precisely, consider for both the *inspection* and *reference* images, the corresponding surface intensity vectors each comprised of the intensity values in a *spatially connected region around that pixel*. A flaw is assumed to be present at the pixel if these vectors differ from each other by greater than a flaw strength vector. Once again, characteristic flaw strength vectors are a function of the flaw type and can be determined by examining images of known and/or calibrated flaws.

Two special cases may be considered depending on the nature of the flaw model vector. The first of these, is the *uniform flaw model* which takes each component of the flaw strength vector to be equal. This model is suitable in cases where the flaw has a uniform appearance across the entire neighborhood (e.g. a spot of paint on a surface). The second is the *peak/adjacent flaw model* which takes all but one component of the flaw vector to be constant with the exception being one single component which has a higher absolute magnitude value than the others. The second type is suitable where the flaw has a strong peak value surrounded by adjacent pixels with smaller but uniform values. This provides a crude approximation to the flaw morphology of micro-meteoroid impact craters where the center of the crater is darker than the rest.

Given the definition of a flaw, the null decision hypothesis $\mathcal{H}_0$ assumes that there is no flaw. The flaw decision hypothesis $\mathcal{H}_1$ assumes that a flaw is superposed onto the *reference* image. In order to determine if a flaw is present, the log likelihood ratio [10] is checked to see if it exceeds the test threshold.

Working out the details in the single-pixel case indicates that, as expected, given compensated images corresponding to *reference* and *inspection* images, the flaw detection can be performed by locating flaws at all pixel locations where the differenced image exceeds a predetermined threshold. For the multi-pixel model, the flaw detection process involves taking weighted sums of the differenced image in a suitable window and comparing these sums to a pre-determined threshold. A sub-optimal version of the detection test can be implemented using morphological erosion operations.

With the appropriate model for the flaw, the theoretical flaw detection performance can be analyzed. The performance is dependent on the distribution of the flaw detection signal under the two competing hypothesis: the Null Hypothesis $\mathcal{H}_0$ and the Flaw Hypothesis $\mathcal{H}_1$. If these distributions do not overlap, then it is possible to pick a threshold parameter for the detection process such that all flaws that occur are detected, and at the same time no false-alarms are generated. However, the distributions of the signal under both hypothesis do overlap because of the nature of the noise in the imaging process, and as a consequence ***for any threshold parameter, there will always be a possibility of missing a flaw and of falsely identifying a flaw.*** The selection of the threshold affects the performance of the system and is a function of the characteristic flaw strength and the noise levels in the system. Too high a threshold will decrease the ***probability of detection*** P_D, while too low a threshold will increase the ***probability of a false alarm*** P_F. This aspect of the performance is captured by providing parametric plots of the P_D versus P_F for various cases. These plots are known as *Receiver Operator Characteristics* (ROC's) from their earlier use in radar target detection. A detail analysis of performance has been conducted using these concepts and presented elsewhere [1].

Errors in mis-registration correction and ambient light compensation can be interpreted as increasing the noise in the image leading to lower detection performance. Residual mis-registration errors induce a change in intensity which can be confused with a flaw. Only translational mis-registration effects are considered here since any mis-registration effects arising due to small angular motion in the image plane may be locally approximated as a translational mis-registration. An analysis shows that the intensity difference at a pixel due to mis-registration may be considered as an additional noise term that adds on to the more typical random noise components present in an image. The presence of mis-registration increases the threshold which must be exceeded before a difference value is considered to be a flaw, and consequently reduces the possible performance. In a similar way, interpolation errors during ambient light compensation can also be interpreted as a noise term distributed over the image. If these noise effects are localized then they have less of an impact on the multi-pixel flaw model likelihood-ratio test than on the single-pixel case. This is because of the averaging inherent in determining the likelihood ratio test in the multi-pixel case.

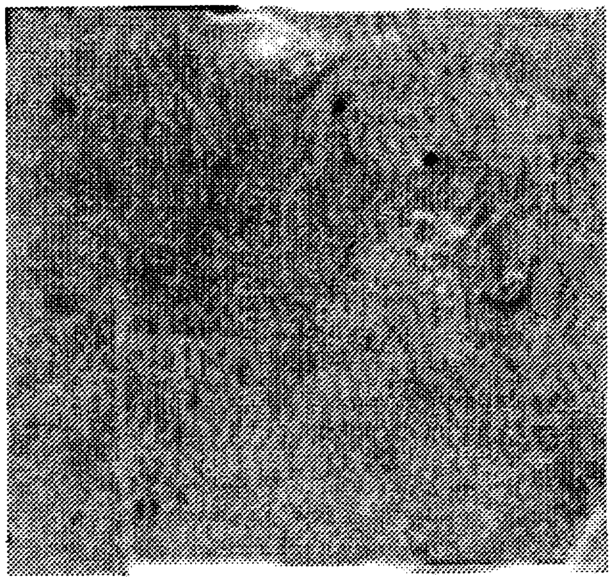


Figure 1: *Large residuals are detected at flaw locations.*

A number of tests under different lighting conditions have been performed to test the flaw detection algorithms. Flaws are simulated by a random dot pattern of a given pixel size distributed on the surface of a test object. Figure 1 shows the final differenced image after mis-registration correction.

2.5 Visual Inspection Summary

The key conclusions are summarized:

- Image differencing appears to be a viable approach for flaw detection with the use of ambient light compensation methods and iterative registration algorithms to overcome the problems of variable lighting and image mis-registration.
- The Gauss-Newton algorithm has been shown to be effective in performing mis-registration correction with large ($\approx$ 10 pixel) registration errors.
- Issues relevant to a flaw-detection theory have been presented and applied to test cases in the laboratory. The quantitative tools developed allow an explicit tradeoff between detection probability and the false-error probability. Depending on the flaw model and noise parameters, detection thresholds can be chosen to achieve a given level of performance.

Areas of further work and necessary improvements are identified:

- *Active Inspection Strategies* need to be developed to improve data collection upon preliminary detection of a potential flaw. The additional data would be used to improve detection performance and could involve commanding additional sensor platform motions to improve lighting and viewing angles; and selection of different illuminator/camera combinations to get more data.
- The information in the ambient lit image needs to be exploited and used to supplement the image information in the compensated image. In the ideal case, the strobe light should be used only to "probe" or supplement the ambient lit image for additional information.
- Multiple imaging with different electronic shutter settings needs to be investigated in order to improve the dynamic range in both bright and dark regions of image.
- Flaw morphology data needs to be captured by supplementing the imaging sensor with a depth/profile sensor.
- Occlusion data needs to be generated at each vista point to allow the anticipation of previously occluded portions of the scene being mistaken for flaws. This might require data from an additional camera or from an additional image taken near each vista point.

3 Integrated Sensor End-Effector

While visual inspection is the primary means of flaw detection, it is only one of the modes available. There are some anomalies, such as errant temperatures and gas leaks, which are not directly detectable by visual information. Therefore, a compact *Integrated Sensor End-Effector (ISEE)* has been developed to house not only the cameras and lights, but a suite of other sensors. Figure 2 shows the recently constructed device, where the labeled components are:

A Two intensity feedback controlled halogen lamps.

B Two fast pulse strobes flashes.

C A parallel jaw gripper.

D Two color cameras calibrated for stereo viewing.

E Two infrared triangulation proximity sensors.

F A six DOF force/torque sensor.

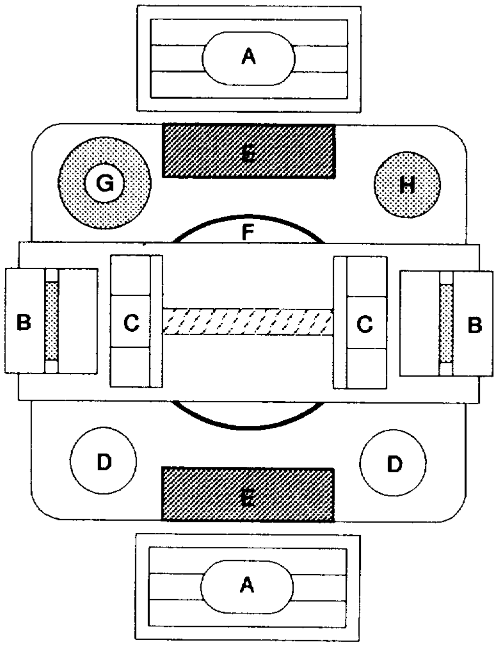


Figure 2: *A front view of the ISEE. The lettered components are a described in the text.*

G An optical pyrometer with a laser sighting beam.

H A Metal Oxide Semiconductor (MOS) gas/vapor sensor.

Proximity sensing is achieved with two infra-red triangulation sensors, sensitive to approximately 0.75 m. The distance measurements are used for collision avoidance, surface contour following, and surface contour measuring. Temperature sensing is achieved with an infra-red optical pyrometer (8-12 micron wavelength), sensitive to temperatures from 0 to 1000°F. Gas sensing is achieved with a multi-gas MOS type sensor which changes resistance as a vapor is absorbed. (While it may be possible to employ this gas sensing technology in orbit, we recognize the superiority of using a compact mass spectrometer in the ambient vacuum of space.)

The controlled lights are maintained at a known illumination level by a optical transistor feedback circuit. This makes the illumination independent of current fluctuations and bulb age, and makes precise measurement and camera characterization possible. This lighting is augmented by extremely compact and fast pulse strobes. The strobes provide short duration lighting of intensity on the order of the Sun but only for short, energy saving, single camera frame, bursts. Since the flashes are mounted on the outside surface of the movable parallel jaws of the gripper, the flash illumination angle may be varied as desired.

All components are commercially available, and have been physically and electrically integrated into the com-

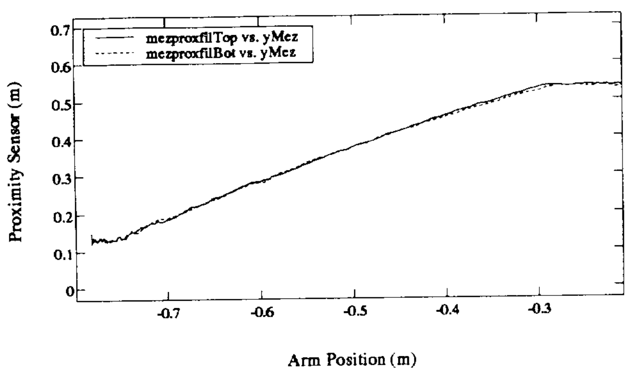


Figure 3: *Experimental data showing the filtered proximity measurements from the two proximity sensors as a function of the arm position. The environment surface was at about -0.81 m.*

pact ISEE end-effector, with a resultant mass of approximately 3.5 kg. The force and proximity sensors, as well as the gripper, are not directly used for inspection. Instead, they aid in the control of the robot arm, and therefore, the end-effector. In particular, the proximity sensors can be used for collision prevention and surface tracking. The development of these capabilities is discussed next.

3.1 Proximity Sensors for Inspection Operations

A demonstration of the utility of proximity sensor environmental position determination for robot collision avoidance has been performed in a real-time implementation. For these tests, two IDEC/Izumi SA1D triangulating range sensor were used [11, 12]. Since the sensors have a minimum sensing distance, they were recessed with respect to a parallel jaw gripper which has a length of 11 cm. The sensor values were read through an A/D board by a 68040 processor (VME bus architecture) at a sampling rate of 44 Hz, and the data was digitally low-pass filtered for noise reduction. Figure 3 shows the filtered readings from the two proximity sensors as a function of the robot end-effector position. The response is fairly linear and consistent between the two sensors.

To use the proximity sensor readings for control of the manipulator, the velocity $\dot{x}_{ps}$ in the block diagram of Figure 4 was commanded as a function of the sensed distance. Two different functions were used: *collision avoidance* and *distance servoing*. Figure 5 shows these two functions, which are identical except for the dashed segments of the servoing function in regions **D** and **E**. The piece-wise continuous formulation was chosen mainly for simplicity in implementation and ease of modification. The value of V_{js} is the maximum velocity

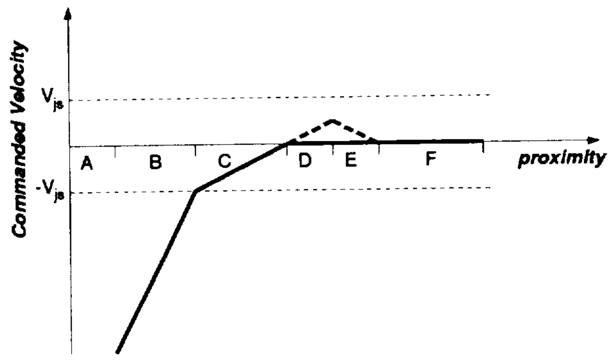


Figure 5: *The piece-wise continuous functions of the commanded velocity $\dot{x}_{ps}$ as a function of sensed proximity. The collision avoidance and distance servoing functions are identical except that the latter has positive values indicated by the long dashed line. See the text for a full description.*

that can be commanded from the joysticks. Operating region **C** provides a collision avoidance velocity command that can not be overridden with a large positive velocity command, $\dot{x}_{js}$, from the joysticks. Operating region **B** allows for quick retreat of the arm if environmental surface protrusions should come into view from the periphery as the the arm is moved tangential to the surface. (It is desirable to restrict the slope and absolute magnitude of the function in this region because of the low sampling rate employed. For instance, had an asymptotic function been employed, there would exist the chance of a very large or rapidly changing velocity command near the asymptote position.) Finally, region **A** will typically never be entered since the sensors are recessed, and the sensor is incapable of determining distances at this range.

Regions **D** and **E** have non-zero values only for distance servoing (the long dash lines in Figure 5). In **D**, the slope is matched to region **C**, to provide equal acceleration to the servo point between **C** and **D**. The peak value of the distance servoing velocity is restricted, to allow negative joystick commands to overcome it and 'pull' the arm away from the surface. Region **E** is provided to make the function continuous. In region **F**, the sensor can detect distance, but the commanded velocity is zero. Outside of **F** the sensor is out of range.

Figures 6 and 7 show the values of $\dot{x}_{ps}$ commanded by the avoidance and servo functions in the real-time implementation. For these measurements $\dot{x}_{ps}$ was not added to $\dot{x}_r$, and a simple linear trajectory away from the environmental surface was used for $\dot{x}_{tg}$.

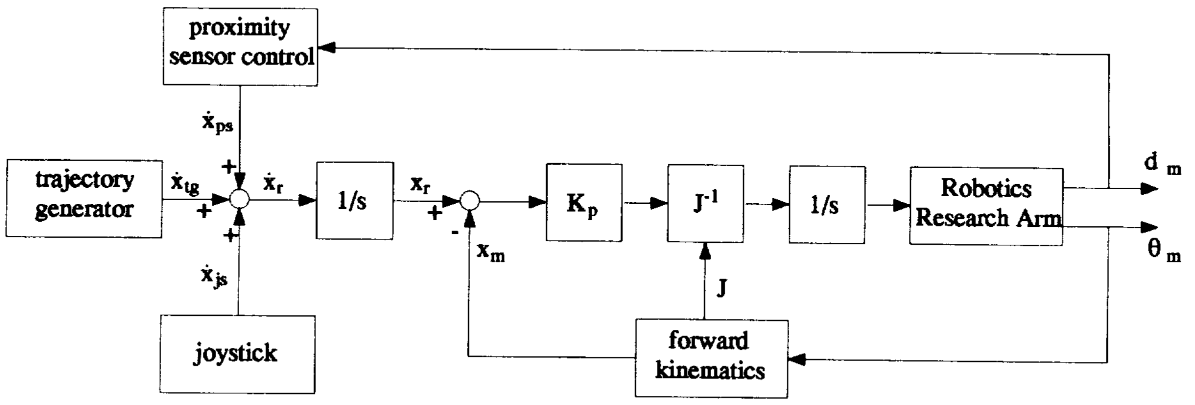


Figure 4: *Block diagram of the control system used for the initial tests of proximity sensor collision avoidance.*

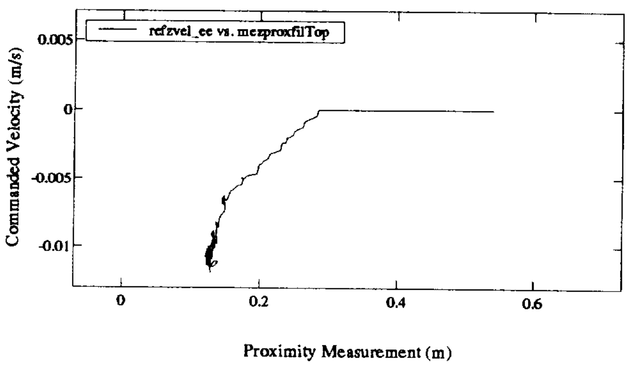


Figure 6: *Experimental data showing the commanded repulsion velocity as a function of measured proximity to the environment.*

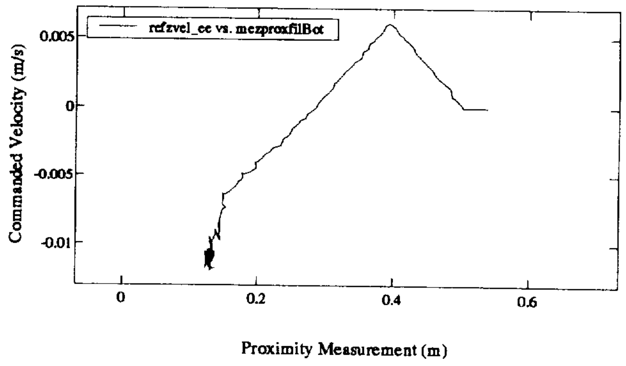


Figure 7: *Experimental data showing the commanded distance servo velocity as a function of measured proximity to the environment.*

4 Orbiter and Sunlight Simulation

To demonstrate the capabilities of the inspection system, a one-third scale mock-up of the Space Station Freedom truss has been created. Figure 8 illustrates the mock-up and its components:

A Electrical Orbital Replacement Unit (ORU) which opens to the left on a hinge to reveal electrical connectors and a Cold Plate.

B Tank ORU.

C Solar Panel.

D Tank and Tubing ORU.

E Simulated hot and cold spots.

F Simulated micrometeor impacts and gas leaks.

The simulated hot and cold spots on the electrical ORU are created using Peltier effect heat pump modules mounted on the inside of the aluminum surface. Since the aluminum has a low emissivity, the outside surface is covered with a circle of Black Kapton to enable the surface temperature to be correctly measured by the optical pyrometer. In the future, the surface temperature may be measured directly by touching it with a thermocouple, eliminating the need for the Kapton.

To introduce a degree of randomness into the inspection process, only two of the Peltier modules are turned on at any time, and the selected direction of electrical current determines if the surface becomes hot or cold. A similar selection is available from amongst the three possible gas "leaks". Each leak uses compressed air to spray a fine mist of household ammonia (to simulate hydrazine) from a small hole on the Tank and Tubing ORU.

Also, random defects may be introduced into the truss mockup through three simple methods. First, screws throughout the truss can be randomly removed to indicate structural defects. Second, small pieces of black tape on pen markings can be placed throughout to simulate micrometeorite impact sites. Third, entire

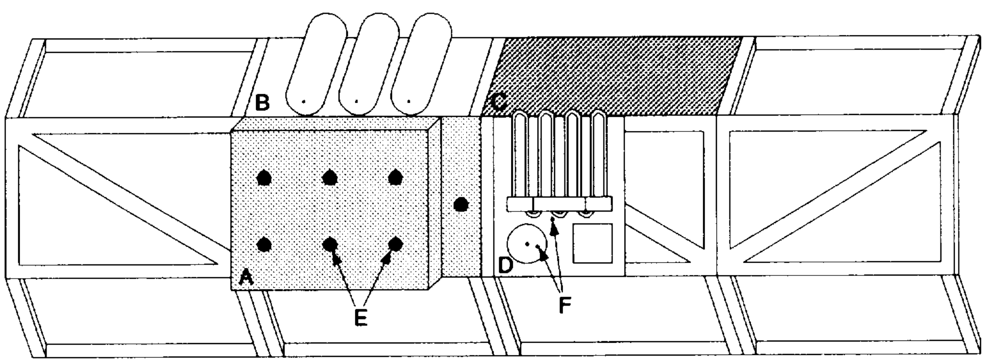


Figure 8: *The mock-up of the SSF truss. The lettered components are described in the text.*

components, such as tanks, can be replaced with defective versions.

Finally, to simulate the space environment around the truss, the mock-up and the inspection robot have been placed in a room darkened by black curtains [4]. The operator can view the mockup and inspection operations from one of three stereo camera views or from a window of the SSF cupola mock-up, in which the inspection station is situated. Simulated sunlight is provided by the Solar Illumination Simulator, discussed next.

4.1 Robotic Lighting Control for Solar Illumination Simulation

Traditional solar simulators are designed for thermal tests of actual spacecraft [5]. To accomplish this, they utilize large vacuum chambers to house the spacecraft, and collimated lighting from arrays of xenon arc lamps. Brightness up to an order of magnitude greater than solar intensity is possible. To test the effects of changing lighting direction, the entire spacecraft is rotated while the illumination remains constant. While this approach is necessary for pre-flight spacecraft testing, it is simply not practical for robotic system prototype development.

Alternatively, we have developed a small scale simulator which effectively mimics the relative motion of the Sun in the sky, while still providing realistically scaled illumination levels [13]. Figure 9 is a photograph of the simulator, a 1500 Watt arc lamp mounted on a four degrees-of-freedom, computer controlled platform. Its ability to pan/tilt/translate, as well as modify the beam shape, enable the illumination angle of the scene to be varied at rates equal to those experienced in low Earth orbit, and maintain a constant illumination flux just as the Sun provides. While the simulated solar illumination is only 1.5% that of true orbital sunlight, Section 2.1 has previously described the compensating adjustments of controlled lighting position, strobe lighting pulses, and camera exposure times, provided by the inspection system [1]. Therefore, the lighting conditions are a realistic test for machine inspection algorithms and human operators.

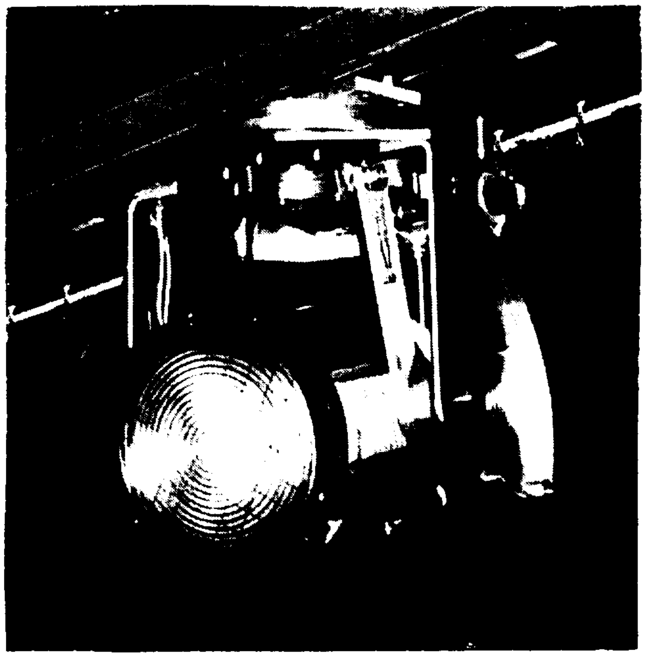


Figure 9: *A photograph of the solar illumination simulation system's robotic hardware.*

Figure 10 shows the solar illumination simulator as a five DOF system, which is represented by its state vector of configuration variables, $\boldsymbol{\theta} = (\rho, \theta, \varphi, \lambda, \gamma)$, where:

ρ, θ, φ	Spherical coordinates from the lamp center to the projected spot center.
λ	Travel of lamp from its origin frame.
γ	Lamp focus parameter indicating position of bulb carriage on internal lead screw.

These parameters have the following ranges:

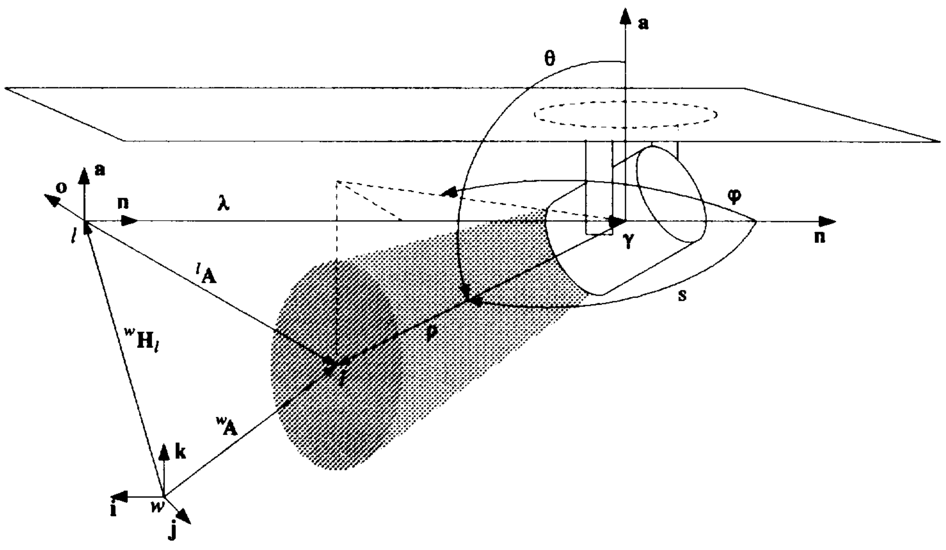


Figure 10: *The configuration and task coordinates for the solar illumination simulator system.*

	MIN	MAX
ρ	0.15 m	∞
θ	90°	180°
φ	60°	120°
λ	0	4.5 m
γ	0	0.076 m

The corresponding task state vector, $\mathbf{x} = ({}^w\mathbf{A}, s, \mathcal{I})$, is composed of the following variables which are also shown in Figure 10:

${}^w\mathbf{A}$	Cartesian vector from world frame to center of projected spot.
s	Beam angle from the lamp frame **n** axis.
$\mathcal{I}$	Light intensity at the center of the spot on the environment.

The task vector is obtained from the configuration vector through the forward kinematics: $\mathbf{x} = \mathbf{F}(\boldsymbol{\theta})$.

Finally, it is important to note that although the kinematics has five DOF, only four are actuated. In the configuration space, the unactuated and unmeasured DOF is the radial distance from the lamp to the surface, ρ. It's value is calculated from the user specified world coordinates, ${}^w\mathbf{A}$. The controller is open-loop for this variable since no real-time measurement of ρ is possible.

In the task space, the unactuated and unmeasured DOF is the light intensity at the surface. Maintenance of the intensity is performed open-loop based on the calculated value of ρ and an optics model which has been experimentally verified [13].

5 Conclusions

This paper has presented the details of some of the technology developed for telerobotic inspection of space platforms such as SSF. Primary amongst the inspection technologies has been visual inspection using computer processing of images from robotically controlled cameras. The processing provides ambient light compensation, registration correction, and automatic flaw detection based on the described flaw models. Secondary inspection and other sensory data are provided by gas, temperature, proximity, and force sensors integrated into the compact ISEE end-effector. This device has been described and the proximity sensor based control of collision avoidance and surface following has been highlighted. Finally, a complete description has been given for the simulated orbiter defects and the space environment lighting. This simulation environment has allowed more rigorous testing of the developed inspection devices and methods.

6 Acknowledgments

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1] J. Balaram. Automated Visual Change Detection For Remote Surface Inspection. Internal Engineering Memorandum 3474-93-004, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, September 1993.

[2] F. Fisher and C Price. Space Station Freedom External Maintenance Task Team — Final Report. Technical report, Johnson Space Center, Houston, Texas, July 1990.

[3] P. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press.

[4] S. Hayati et al. Remote Surface Inspection System. *Journal of Robotics and Autonomous Systems*, 11(1):45–59, 1993.

[5] Our Captive Space: JPL Space Simulator Facilities. Brochure 400-68, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, September 1993.

[6] W. S. Kim. Graphical Operator Interface for Space Telerobotics. In *IEEE International Conference on Robotics and Automation*, pages 761–768, Atlanta, Georgia, May 2-6 1993.

[7] L. Murr and Kinard W. Effects of Low Earth Orbit. *American Scientist*, 81-2:152–165, 1993.

[8] T. See et al. Meteoroid and Debris Impact Features Documented on the Long Duration Exposure Facility. Technical Report 24608, Johnson Space Center, Houston, Texas, August 1990.

[9] H. Seraji, M. Long, and T. Lee. Motion Control of 7-DOF Arms: The Configuration Control Approach. *IEEE Transactions on Robotics and Automation*, 9(2), April 1993.

[10] H. Van Trees. *Detection, Estimation, and Modulation Theory: Part 1.* John Wiley.

[11] R. Volpe. Year End Task Report of The Prototype Safety System for Robots Near Flight Hardware. Internal Engineering Memorandum 3474-93-006, Jet Propulsion Laboratory, Pasadena, CA, October 1993.

[12] R. Volpe. A Survey and Experimental Evaluation of Proximity Sensors for Space Robotics. In *submission to The IEEE International Conference on Robotics and Automation*, May 1994.

[13] R. Volpe and D. McAffee. A Robotic Lighting System for Solar Illumination Simulation. In *submission to The IEEE International Conference on Robotics and Automation*, May 1994.

5/ - 63
207642
6040

AIAA-94-1194-CP

A HIGHLY REDUNDANT ROBOT SYSTEM FOR INSPECTION

N94-30545

p. 7

Thomas S. Lee, Tim Ohms, and Samad Hayati
lee@telerobotics.jpl.nasa.gov
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California, 91109

Abstract

The work on the serpentine inspection system at JPL is described. The configuration of the inspection system consists of 20 DOF in total. In particular, the design and development of the serpentine micro-manipulator end-effector tool which has 12 DOF is described. The inspection system is used for application in JPL's Remote Surface Inspection project and as a research tool in redundant manipulator control.

1. Introduction

For several years, the Jet Propulsion Laboratory (JPL) has been performing research and development in remote surface inspection of space platforms such as Space Station Freedom [1]. One of our goals was to develop technology to inspect remote, hard-to-reach locations. Our experimental facility contains a 1/3-sized mockup of the Space Station truss structure with various devices attached. The structure is cluttered with different types of objects such as an Orbital Replacement Unit (ORU) and a thermal radiator. The tasks to be performed range from visual inspection by maneuvering inside of narrowly confined areas and detecting anomalies to temperature and gas leak detection. One such scenario is moving behind a radiator panel and searching for electrical damages. Others include detection of broken interfaces such as disconnections in fluid, gas (leaks), or electrical lines and improper mating of connectors. There are some light manipulation tasks which are required to diagnose, service, and repair devices attached to the space structure. Some of the manipulation tasks include spot cleaning, foreign object debris location and removal, and removal/installation of straps and caps for lenses or containers.

Conventional robots typically consists of 6 Degrees-of-Freedom (DOF), and are not capable of performing some of the required remote inspection tasks. At JPL a highly redundant robot inspection system consisting of 20 DOF will be utilized. The idea is to attach a smart end-effector tool that has a long-reach serpentine feature at the end of a conventional robot. This arrangement is referred as a *compound robot* — the serpentine robot is the *micro-manipulator*, and the base robot is the *macro-manipulator*. Figure 1 shows this configuration. Note that the 7 DOF of Robotics Re-

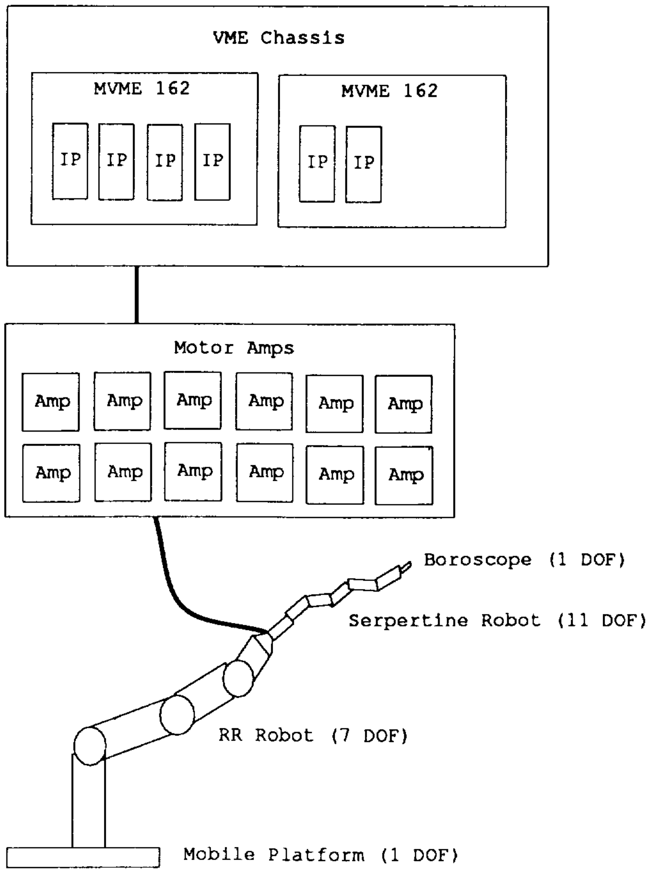


Figure 1: Overall Inspection System and the Hardware Architecture

search arm is mounted on a 1 DOF mobile base. The macro-manipulator can be thought of as a global positioning device, while the micro-manipulator can be viewed as a fine manipulator restricted to operate in a local region. In this paper, the design and development of the serpentine micro-manipulator is described. (see Figure 2).

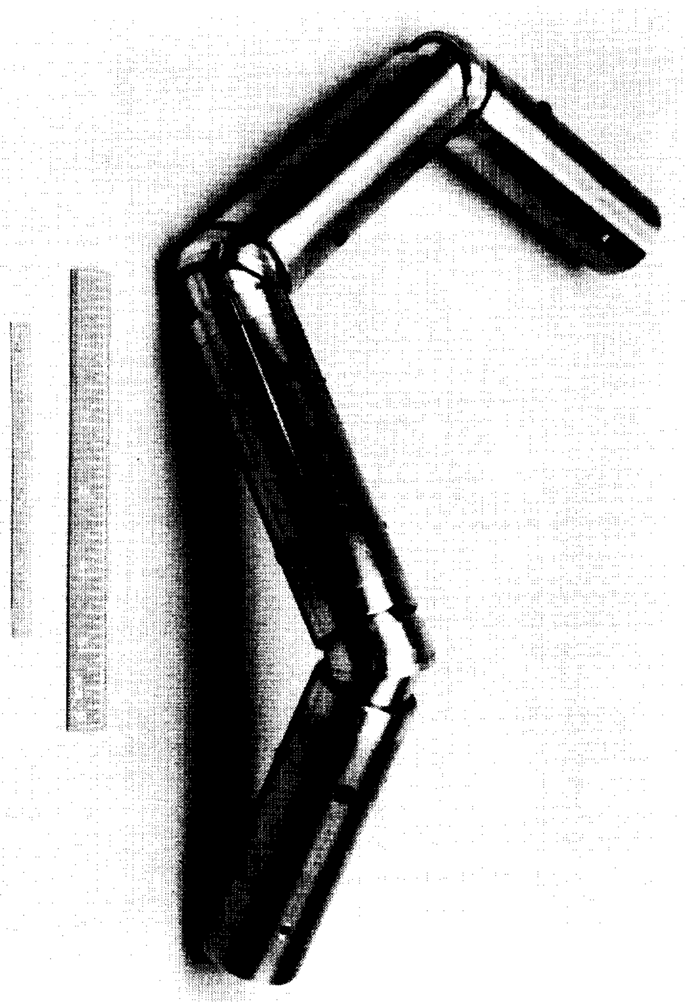


Figure 2: The JPL Serpentine Robot

2. Background

Work in serpentine robotics dates back approximately 30 years. Namely, the Japanese companies such as Toshiba, Mitsubishi, and Hitachi have done a lot of work in this area for application in the nuclear power industry. Hirose [2] of Tokyo Institute of Technology developed a number of snake-like mechanisms, for example, a crawling mechanism which utilizes oblique swivel joints. Asano [3] built Toshiba's Self Approach System in 1982. A camera was mounted on the tip of this 16 DOF tendon-driven mechanism to perform inspection. In the Unites States, notable works include Anderson and Horn [5] who built a 16 DOF tensor arm for Scripps Institute of Oceanography in 1964. Chirikjian and Burdick [6] of Caltech built a 30 DOF variable geometry truss manipulator to validate hyper-redundant arm control algorithms. Berka [7] performed research in multi-segment robots for NASA's Johnson Space Center.

3. Serpentine Robot Design

At the end of the macro-manipulator, an integrated sensor/end-effector (ISEE) unit is attached [4]. It contains 2 lipstick cameras, 2 proximity sensors, a gas sensor, a temperature sensor, a force/torque sensor, and two light fixtures. This unit is too bulky to enter inside of the mockup truss structure. To overcome this restriction, a serpentine robot that can function as a smart end-effector tool was designed. The serpentine robot would be picked up by the macro-manipulator when additional dexterity is required to perform the task.

A number of design issues were considered before building the serpentine robot. The issues and their resolutions are discussed as follows.

A. Weight and Size

Since the serpentine robot is to be attached at the end of another robot, weight and size needed to be minimized.

Motor Selection: Miniature, yet high torque motors were needed. Motor manufacturers such as Escap, Maxon, and MicroMo were considered. MicroMo's 2 watt DC motors were chosen. Based on ironless core technology, these products have the feature of high efficiency with low mechanical time constants. The motors have stall current of 890 mA, and due to their low inductance, electrical noise is reduced.

Joint Assembly: The joint design needed to be compact. If the conventional method of mounting the motors on the joints were adopted, the serpentine robot would have had a bulky design. A patented design owned by the NEC Corporation was chosen. This design allows all motors to be mounted inside of the joint housings.

The original design is an active universal joint based

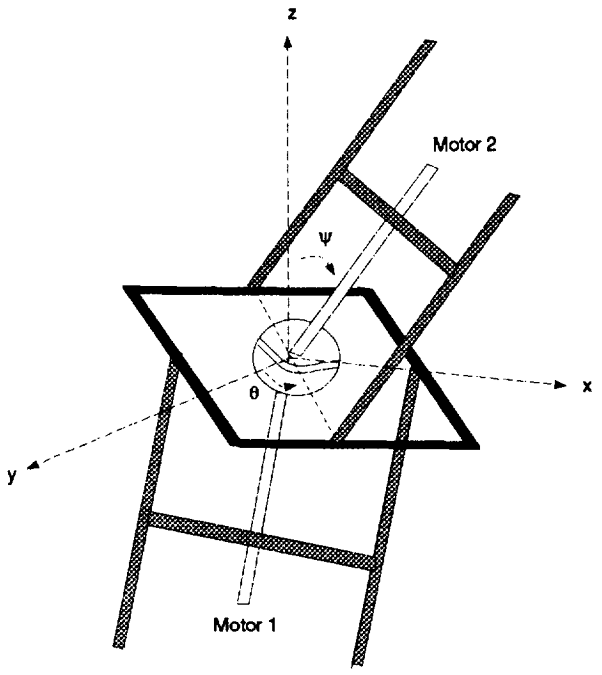


Figure 3: Joint Assembly

on work by Ikeda and Takanashi of the NEC Corporation (U.S. Patent No. 4,683,406). Our mechanism was made more compact by modifying their design. The basic idea is illustrated in Figure 3. The joint assembly has two shafts, with each shaft attached to a half-sphere at an oblique angle. The two half-spheres are joined together to rotate freely with respect to each other. This arrangement is contained inside a universal joint with each shaft joined to one side of the frames that make up the universal joint. The motors rotate the two shafts thereby actively changing the orientation of the universal joint. Both motors are controlled simultaneously to change the orientation. Now consider the Spherical coordinate system. When the motors are rotated in the same circular directions, the joint assembly makes a motion along the ψ direction. If the motors are rotated in opposite directions, then the joint assembly makes a motion along the θ direction. The motions along the ψ and θ directions make up the 2 DOF movement of the joint. Note that when the shafts are collinear, a degeneracy (singularity) occurs.

To achieve high torque, each axis has a gearhead ratio of 1111:1 (high gear ratio was achieved by building our own custom planetary stages). Two redundant motors which are mechanically coupled turn each axis and provide double the torque of one motor. The geartrain is non-backdriveable for reduced power consumption. Maximum torque at each DOF was theoretically computed to be 90 in-lb, which was experimentally verified. Figure 4 and Figure 5 reveal the internals of the joint assembly.

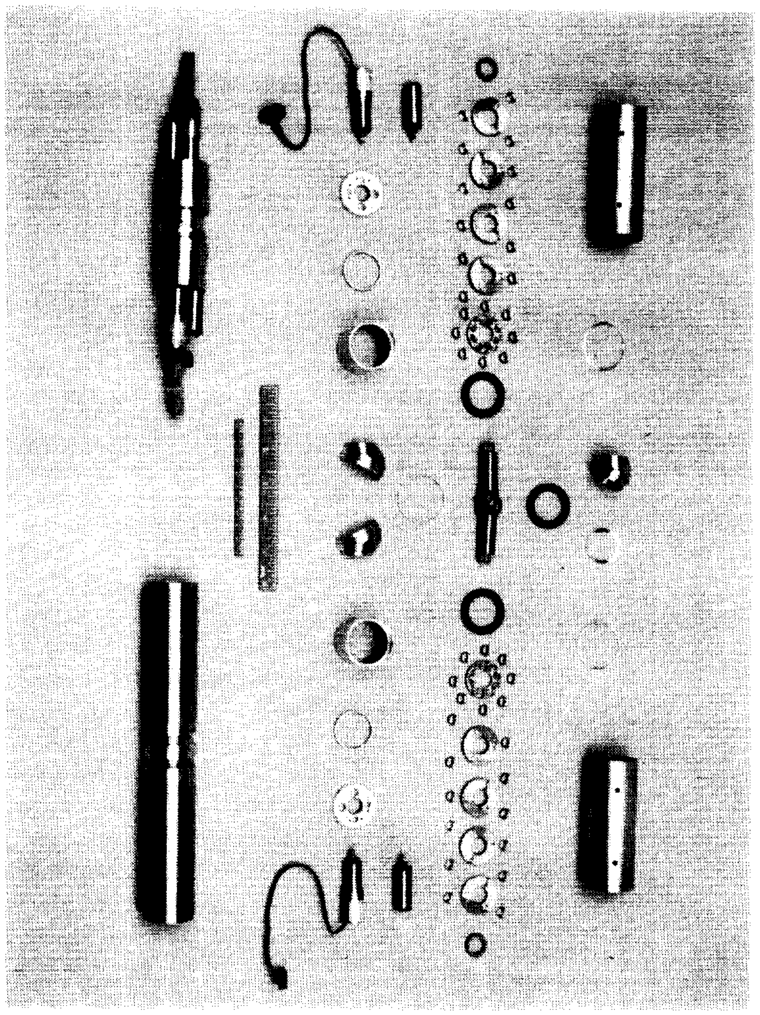


Figure 4: Components of the Serpentine Joint

B. Reliability and Ease of Control

To reduce the size and weight, building a tendon-driven mechanism was considered. This approach is appealing because the actuators can be moved to the base of the serpentine robot. Since the entire serpentine robot including its base needed to be picked up by another robot, the overall mass is not saved by using this approach. In addition, inherent difficulties exist in dealing with a complicated tendon mechanism. This type of mechanism typically has a small load capacity, and it is difficult to model. Problems exist because of

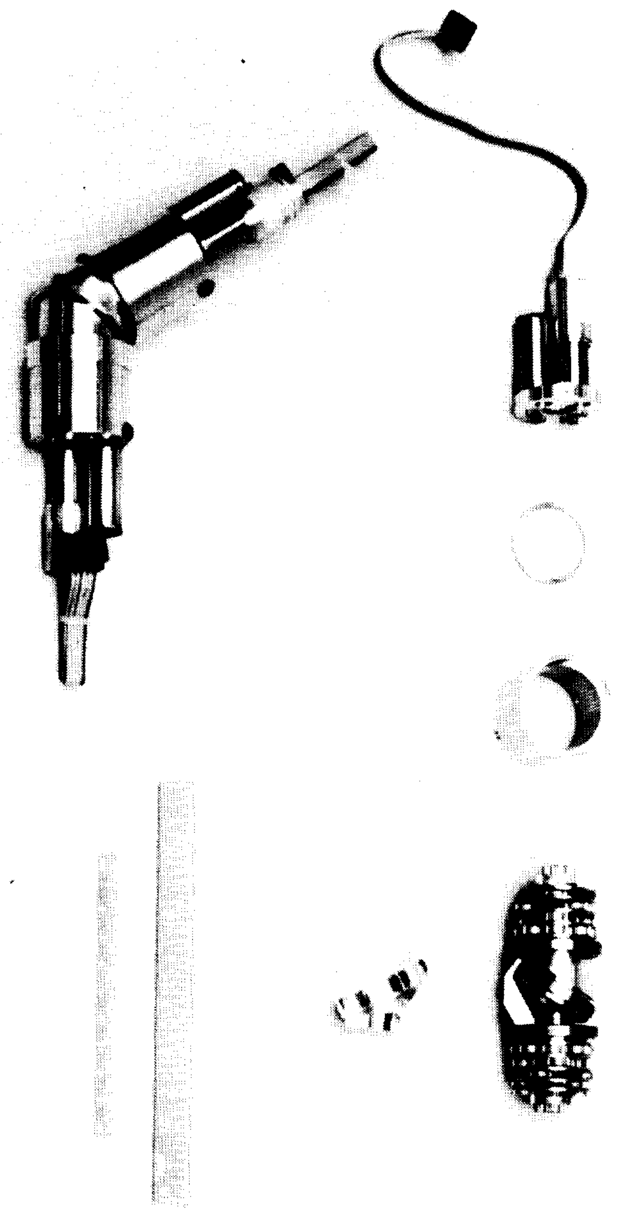


Figure 5: Internals of the Joint Assembly showing the Planetary Stages

the need for flexible control to compensate for elasticity. Finally, low reliability results due to frequent tendon breakage.

A method of direct motor control was chosen. Although the problems associated with high gear ratio will have to be dealt with, better reliability would be obtained.

C. Modularity

The mechanism was designed to be mechanically modular — the joints can be easily added or subtracted. The concern was more on the electrical side.

Designing miniature circuits to fit inside of the joint housing was considered. The electronics would provide the functionalities of a motor amplifier and a decoder for encoder signals. In designing a linear amplifier, elimination of heat generated by the electronics would create a problem since insufficient volume exists for air ventilation. Even a cooler PWM-based amplifier that employs miniature H-bridges could not be contained, since the size of all of its electronics would exceed the size of the joint housing (a cylinder of 1.5 inches in diameter with 5.65 inches height). To generate control signals, commercially-available controllers such as the NEC uPD7832x, Hewlett Packard's HCTL-1100, and LM628 chips were considered. Circuit designs based on any of these chips would exceed the size of the joint housing.

The option to route all the wires out of the robot was chosen. The motors will be controlled remotely from externally located VME hardware. Routing all wires internally through the center hole posed another problem — cabling. Because 23 motors exist inside of the serpentine robot, the number of through-hole wires had to be minimized. The wire count was reduced at each DOF by connecting two motors in parallel to share motor voltage lines and by sharing common power lines for all motors. See Figure 6 for the wiring diagram.

For external VME control of the motors, off-the-shelf hardware were purchased. Because of the motor's low inductance, linear analog amplifiers rather than PWM types were chosen as motor drives. Motor controller hardware were purchased to work in the VMEbus environment.

D. Acquiring Visual Data and Lighting

Mounting a small lipstick camera (e.g., Toshiba Model IK-M41A) at the tip of the serpentine robot was considered. This approach has associated problems with wiring and lighting. The diameter of the camera's cable far exceeds the size of the through-hole. Furthermore, the standard way of providing light for the camera is to resort to installation of light fixtures. But since the light fixtures are typically larger than the lipstick cameras, the size advantage of using the miniature cameras would be lost.

Using a borescope was ideal for our purpose. A borescope is designed specifically for visual inspection applications. It is commonly used in medical surgeries and aircraft engine inspections. The video image of the work site is passed through its fiberoptic cable and is sent remotely to the viewer — most of the vision hardware is located away from the robot's end-effector, hence moving the bulkiness away from the work site.

The Machida FBA-3-140 flexible borescope was chosen. The fiberoptic cable has a diameter of 0.138 inch (3.5mm) and 55 inches long. With a through-hole of 0.312 inch (5/16 inch), both the borescope cable and required control wires were routed internally. The scope has a field of view of 50 degrees minimum and a depth of field of 5 to 50 mm.

The borescope is capable of 1 DOF motion. The tip is articulated by manually moving the lever at the eyepiece which pulls the cables attached to the tip. It is capable of a range of motion from -100 to 100 degrees. The function of the lever was motorized by installing a motor at the base of the serpentine robot to pull the cables. A working channel can be mounted along the side of the borescope to allow remote use of small tools, for instance, a grasping tool to retrieve foreign objects and a grinding tool to smooth surfaces. A working channel may be installed in the future to perform simple manipulation tasks.

An advantage of using a borescope is it carries its own light. When the serpentine robot enters the inside of the space structure, the environment is typically dark. Therefore, to acquire visual images, lighting is required. With the borescope, lighting is built into the cable and points in the same direction as the head of the borescope. Since our mockup structure composed mostly of metals with high reflectance, minimal light for the borescope was required — a Halogen light source served our purpose.

One drawback of using a borescope is it cannot by itself bore into the work area. A common way is simply pushing the borescope to insert it into the work area. To assist in the boring operation, for example in medical application, guide tubes are available to make possible insertion into difficult places where obstructions or large gaps exist. The guides are contouring apparatus to make angled turns possible by conforming to the desired insertion path. Here the serpentine robot can be thought of as a flexible guide tube for the borescope. The serpentine robot will act as a contoured platform for the borescope to rest on while the operator looks around the work area.

E. Mechanical Specifications

Constructed serpentine robot has the following specifications:

- 3-D Mechanism with Total Weight of 7 lbs
- Extended Reach: 35"
- Diameter of the Robot: 1.5"
- 5 Joints, 10 DOF (each $-60°$ to $60°$)
- 1 Roll DOF ($-180°$ to $180°$)
- 1 Borescope DOF ($-100°$ to $100°$)
- DOF Velocity : 60 degrees/second
- Center-to-Center Joint Distance: 5.65"
- Through-Hole Inside for Cables: 5/16"

F. Macro-manipulator

The larger manipulator is the Robotics Research Corporation's Model K1207 robot which has 7 DOF. This arm is mounted on a mobile platform of the lathe-bed and provides one additional prismatic DOF. In total, 8 DOF comprise the macro-manipulator.

G. User Interface

The operator will interface with the serpentine robot from the "cupola," which is the main control station of the experimental facility of the Remote Surface Inspection project. Inside the cupola, one has access to an IRIS Silicon Graphics workstation, color monitors, and joysticks. The IRIS will act as a graphical front-end through which the operator interacts with the serpentine robot in real-time and issues motion commands in joint or task space. The IRIS can also create an interactive graphical simulation environment for analysis and control of the serpentine robot. Using this dual-mode functionality, the IRIS can be used in preview mode for animating the task scenario, followed by commanding the arm to duplicate the simulated motion.

The operator will view the work site by looking at the monitors that display video images from the borescope, and he will command the serpentine robot by using the joysticks and a graphical menu on the IRIS.

4. Serpentine Robot Control System

Industry Pack (IP) Servo modules from Technology 80, Inc. are used to control the motors in a VMEbus environment. These units are built around National Semiconductor's LM628 ICs and provide 2 independent channels for PID motor control and decoding of encoder signals. The IP-Servo modules are mounted on MVME162 Motorola processor boards which are based on the MC68040 hosts running at 25 MHz. See Figure 1 for the hardware architecture. To control the serpentine robot, two Motorola processor boards are employed to host six IP-Servo modules. The two processor boards are plugged into the same VME chassis that provide VME control for macro-manipulator system [8]. Through a shared memory card, command and status information of the serpentine robot are passed to the macro-manipulator system. All of the software executing on the VME environment is written in the C language. Code is developed on a SUN UNIX computer utilizing its resident C compiler and Wind River's VxWorks/Wind real-time library.

The IP-Servo module produces motor control signals in the form of voltages. The control signals are then taken as input to a linear analog amplifier. Portescap's ELD-3503 was chosen. This unit is a transconductance type of amplifier which is specifically designed to drive ironless motors. It produces up to 2.5 Amps of current and drives up to 35 Volt motors with a single DC power supply.

5. Future Work

In the near future, kinematic analysis will be performed to achieve Cartesian control of the serpentine system. In the process, a scheme to resolve redundancy of the mechanism would have to be devised to allow a task to be performed by allowing cooperation between the macro- and micro-manipulators. One possible scenario is to allow cooperation between the two manipulators to avoid obstacles by having each manipulator to executing a separate redundancy resolution scheme with a different objective function. Second, control experiments will be performed and any instability problems will be resolved. Problems associated with high gear ratios may exist, and instability may be attributed to the joint assembly since the joint angles are indirectly controlled by motor angles.

Many practical issues need to be dealt with before a three dimensional serpentine robot can be used for a teleoperation task. The manipulation task is difficult, since the operator is maneuvering the robot inside a narrow-spaced workspace and the objects that are of interest to him are often visually obstructed.

Sensors are crucial in helping the operator to perform inspection. The borescope inside of the serpentine robot will provide the main visual feedback to the operator. An additional camera can be attached to one of the intermediate links of the serpentine robot to provide the operator with a wider view of the work area from a different perspective. Other sensors such as proximity sensors can be used to detect and avoid obstacles.

The tip of the borescope should be placed such that it is jitter-free (statically stable) to take still images and to be optimally positioned for collision avoidance. In this scenario, the *active perception* problem of moving the cameras (sensors) would have to be examined to obtain more information about the environment as the task progresses.

The system requires a man-machine interface capability to control the motion of the micro- and macro-manipulators collectively or individually, control the viewing angles attached to the serpentine robot, and ability to work with a world model of the environment for collision avoidance.

Knowledge-based systems can be integrated into the inspection system. In order to guide the serpentine robot, the computer can assist the operator in controlling the camera viewing and lighting angles. Once the operator selects an object/feature, the system can automatically adjust the camera viewing angle (aligning to the normal of the surface and to have the greatest visibility) as well as the lighting angle and intensity for the best view.

In addition, being preoccupied with a difficult teleoperation task at hand, the operator should not have to be concerned about kinematic anomalies such as singularities and joint limits. The operator needs only to specify the trajectory of the head of the serpentine robot; the trajectory of the rest of the body should be computed autonomously with some guidelines from the operator.

All of the above requirements can be incorporated into a global scheme to resolve the kinematic redundancies of the micro- and macro-manipulators.

6. Acknowledgements

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We would like to thank Nobuaki Takanashi for providing the information on the patented mechanical design.

References

[1] Hayati, S., et. al, "Remote Surface Inspection System," *Robotics and Autonomous Systems*, Vol 11, No. 1, pp. 45-59, May 1993.

[2] Hirose, S. et. al., "Design and Control of a Mobile Robot with an Articulated Body", *The International Journal of Robotics Research*, Vol. 9, No. 2, April 1990.

[3] Asano, K., et al., "Multijoint Inspection Robot," *IEEE Transaction on Industrial Electronics*, Vol IE-30, No. 3, pp. 277-281, 1983.

[4] Volpe, R., Balaram, J.B., "Technology Development for Robotic Surface Inspection," *Proceedings of the AIAA/NASA Conference on Intelligent Robots for Factory, Field, Service, and Space*, Houston, Texas, March 21-24, 1994.

[5] Anderson, V., Horn, R.C., "Tensor-arm Manipulator Design," *ASME Transaction*, Vol. 67-DE-57, pp. 1-12, 1967.

[6] Chirikjian, G., Burdick, J., et. al, "Kinematics of Hyper-redundant Robot Locomotion with Applications to Grasping," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991.

[7] Berka, R., "Development of a Large Space Robot: A Multi-Segment Approach," NASA Johnson Space Center Internal Publication.

[8] Seraji, H., Long, M., Lee, T., "Motion Control of 7-DOF Arms: The Configuration Control Approach," *IEEE Transactions of Robotics and Automation*, Vol. 9, No. 2, April 1993.

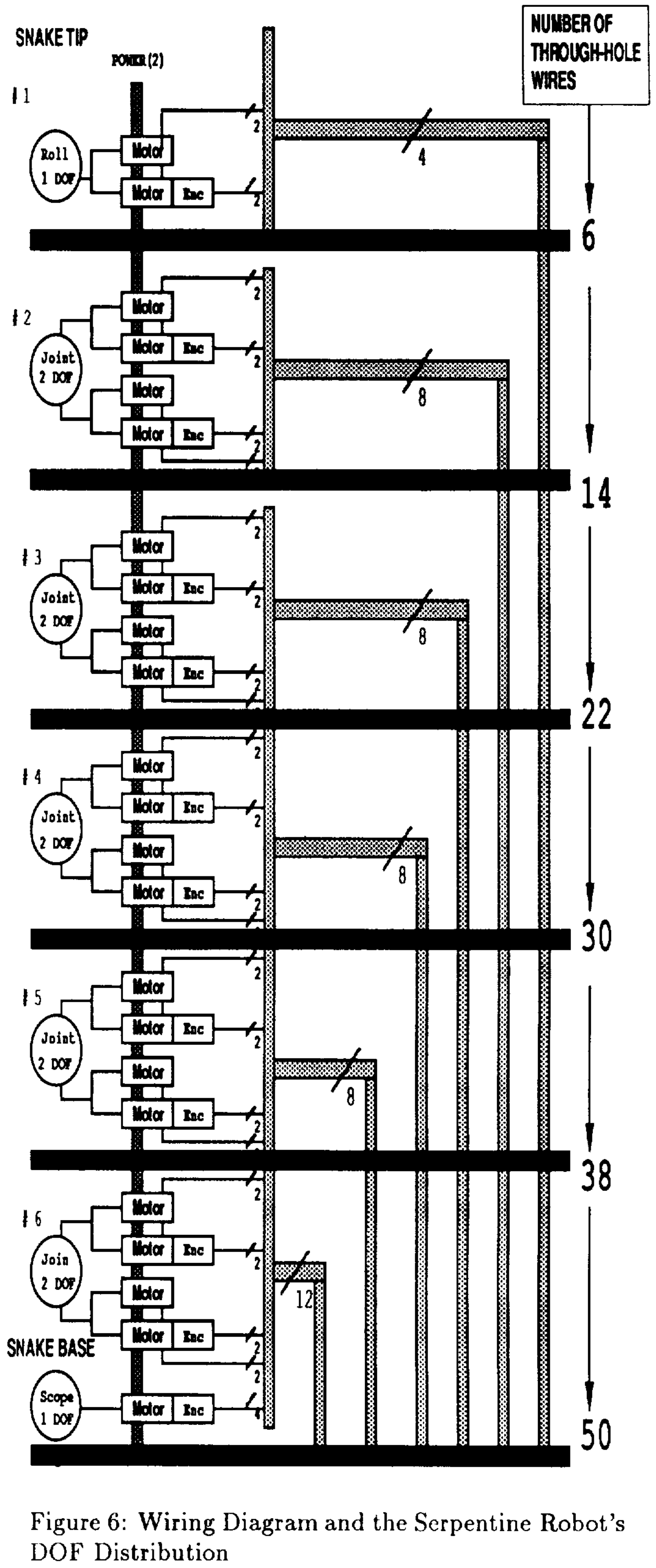


Figure 6: Wiring Diagram and the Serpentine Robot's DOF Distribution

604

N94- 30546

AIAA-94-1195-CP

Free-floating Dual-arm Robots for Space Assembly

S20-63

207543

P- 9

Sunil Kumar Agrawal, Asst. Professor
M.Y. Chen, Graduate Student

Department of Mechanical Engineering
Ohio University, Athens, OH 45701.

Abstract

Freely moving systems in space conserve linear and angular momentum. As moving systems collide, the velocities get altered due to transfer of momentum. The development of strategies for assembly in a free-floating work environment requires a good understanding of primitives such as self motion of the robot, propulsion of the robot due to onboard thrusters, docking of the robot, retrieval of an object from a collection of objects, and release of an object in an object pool. The analytics of such assemblies involve not only kinematics and rigid body dynamics but also collision and impact dynamics of multibody systems. In an effort to understand such assemblies in zero gravity space environment, we are currently developing at Ohio University a free-floating assembly facility with a dual-arm planar robot equipped with thrusters, a free-floating material table, and a free-floating assembly table. The objective is to pick up workpieces from the material table and combine into prespecified assemblies. This paper presents analytical models of assembly primitives and strategies for overall assembly. A computer simulation of an assembly is developed using the analytical models. The experiment facility will be used to verify the theoretical predictions.

1 Introduction

Over the last two decades, a number of studies have been reported on motion planning of free-floating robots ([10], [7], [12], [13], [9], [11], [1], [2], [3], [4]). However, none of these studies dealt with analytics of entire assemblies in a free-floating work environment using free-floating robots. The analytics of these assemblies involve not only kinematics and rigid body dynamics but also collision and impact dynamics of multibody systems. In an effort to understand assemblies in zero gravity space environment, we are currently developing at Ohio University a free-floating assembly facility with a dual-arm planar robot equipped with thrusters, a free-floating material table, and a free-floating assembly table. The objective of this experiment testbed is to verify the analytics of assemblies in free-floating work environment. This paper is organized in the following way: An outline of the free-floating robot facility of Ohio University, its analytical descriptions, and kinematics are presented in Section 2. The analytical models of the assembly primitives such as self motion, propulsion, docking, pickup, and release are described in Section 3. An assembly problem is discussed in Section 4. An outline of a general purpose simulation program FLOAT is described in Section 5 which is designed to study strategies of assembly.

2 Free-Floating Facility

2.1 Physical Setup

The free-floating robot facility of Ohio University consists of a free-floating dual-arm planar robot, a free-floating material table, and a free-floating assembly table. A photograph of the dual-arm free-floating robot is shown in Figure 1. Each of these three units rests on a granite surface supported by four air bearings. Regulated supply of Nitrogen from pressurized cylinders float the units on the granite surface. The robot consists of two arms, each with 3 revolute joints and a prismatic joint. The 3 revolute joints provide the end-effector full mobility in the plane. The prismatic joints are used to move the arms normal to the table. The 8 joints are driven by dc servomotors fitted with optical encoders. A PC 386 motherboard with power from rechargable lead-acid batteries sits on the base of the robot. The motherboard is connected to an 8-axis motion control board and a DAS board. Two quad-thrusters are mounted on the base which are controlled by solenoid valves that use air supply from the tank [5]. The robot communicates with a host PC 486 workstation through a radio-wave modem. Two light bulbs fixed on the base of the robot are tracked by an overhead optoelectronic sensor

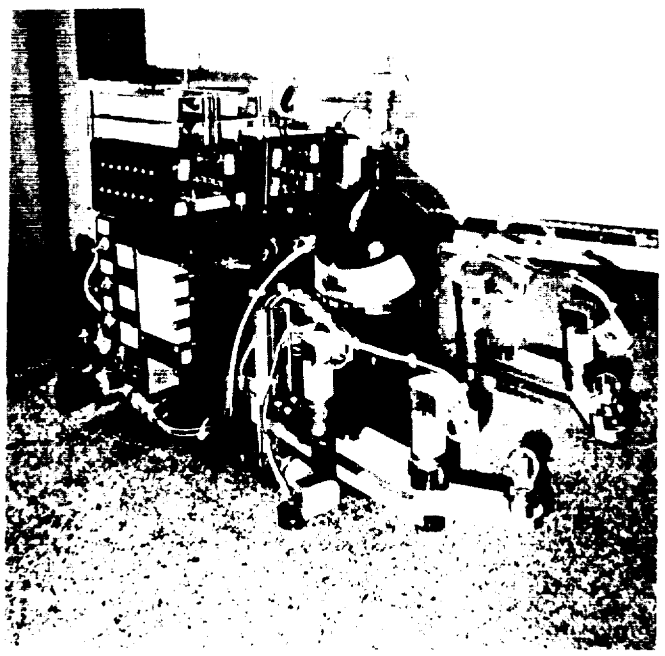


Figure 1: A photograph of a free-floating dual-arm planar robot built at Ohio University.

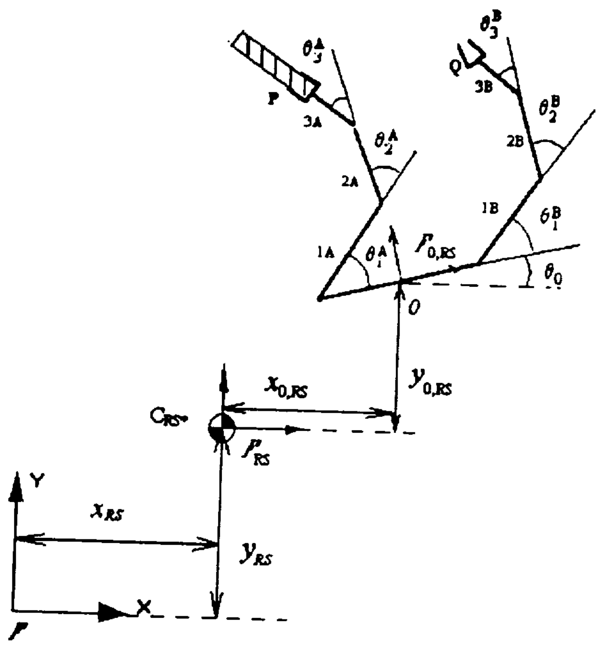


Figure 2: An analytical model of a dual-arm free-floating planar robot.

consisting of a Position Sensitive Detector (PSD) fixed at the focal plane of a TV lens [6]. The PSD sensor is connected to a host PC 486 computer and the voltage outputs of the sensor are calibrated to the position of light bulbs on the table. The sensor provides a feedback of base position and orientation.

The material table also has a pressurized Nitrogen tank which provides the gas needed to float the table on the granite surface. The material table has two light bulbs which are used to feedback the position and orientation of the table to the host PC 486 computer. This table has polished grooves to place the work pieces for assembly. The assembly table has a setup for floatation and position feedback similar to the material table. The grooves in the assembly table are designed to store subassemblies and final assemblies.

One of the assumptions made in this paper is that the joints of the robot are locked during propulsion, docking, pickup, and release and are unlocked during self motion.

2.2 Analytical Modeling

From an analytical standpoint, the free-floating facility consists of the following three systems: (i) the robot system (RS), (ii) the material table system (MS), and (iii) the assembly table system (AS). These three systems are made up from the following units: (i) the dual-arm robot, (ii) the material table, (iii) the assembly table, and (iv) the individual work pieces (W_i). The definitions of these three systems change as the assembly progresses and the workpieces are passed from one system to the other.

2.2.1 Robot System

The robot system (RS) consists of the robot and workpieces held by the end-effectors. The robot consists of seven links and its two arms are labeled as A and B. The base is labeled as 0, the three links of arm A are 1A, 2A, and 3A, and the three links of arm B are 1B, 2B, and 3B. The gripper points on the end-effectors of A and B are respectively P and Q. These grippers are designed to catch the workpieces so that they extend outwards from the end-effector links. The joint angles of arm A are θ_1^A, θ_2^A, θ_3^A and of B are θ_1^B, θ_2^B, and θ_3^B. The prismatic joints in the two arms are not modeled because they are used only periodically to lower and lift the end-effectors. A coordinate frame $\mathcal{F}$ is fixed inertially to the granite table parallel to the edges. A coordinate frame $\mathcal{F}_{RS}$ is fixed at the center of mass of the robot system C_{RS*} with axes parallel to the axes of $\mathcal{F}$. $\mathcal{F}_{0,RS}$ is fixed on the base link at the midpoint of the two joints located on it. The origin of $\mathcal{F}_{RS}$ is described relative to $\mathcal{F}$ by the coordinate variables x_{RS} and y_{RS}. The coordinates $x_{0,RS}$ and $y_{0,RS}$ describe the origin of $\mathcal{F}_0$ relative to $\mathcal{F}_{RS}$. θ_0 is the relative orientation between the X axes of $\mathcal{F}_{0,RS}$ and $\mathcal{F}$. Each link has a mass m_i^j, a center of mass C_{i*}^j, and a moment of inertia I_i^j for $i = 1,2,3$ and $j = A, B$. These quantities for the base link are respectively m_0, C_{0*}, and I_0. The robot system is shown in Figure 2. During assembly, the inertial

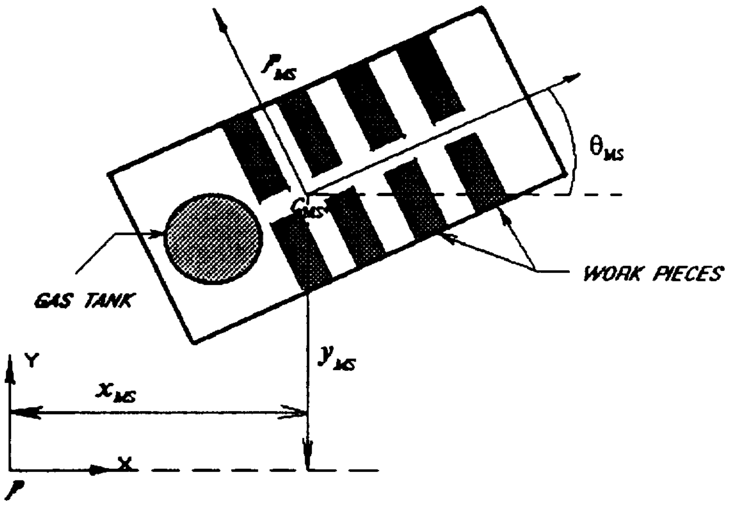


Figure 3: A sketch of the free-floating material table.

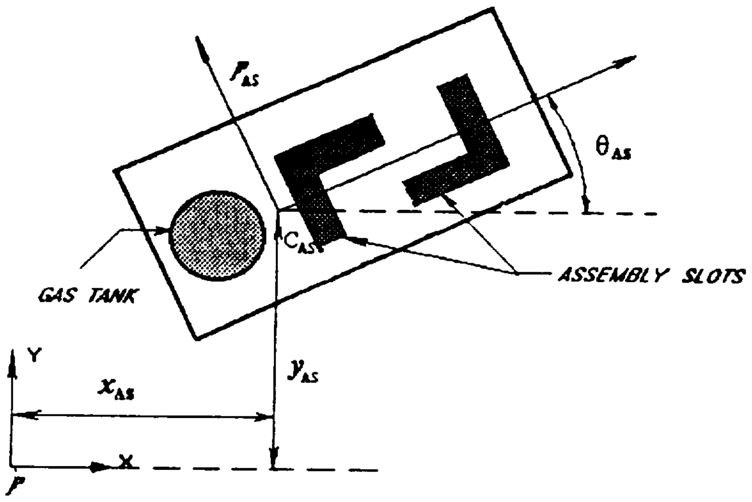


Figure 4: A sketch of the free-floating assembly table.

parameters of the links 3A and 3B are computed from the current definition of the robot system.

In summary, the robot system is described by 11 variables: x_{RS}, y_{RS}, $x_{0,RS}$, $y_{0,RS}$, θ_0, θ_1^A, θ_2^A, θ_3^A, θ_1^B, θ_2^B, and θ_3^B. During self motion, the 6 joint angles of the robot are actively controlled and during propulsion, docking, and pickup, these 6 joints are locked.

2.2.2 Material Table System

The material table system (MS) has 8 slots for the workpieces W_1,...,W_8 to rest. The center of mass of the current system is labeled as C_{MS*}. A coordinate frame $\mathcal{F}_{MS}$ is fixed to MS at C_{MS*} parallel to the edges of the material table. The origin of $\mathcal{F}_{MS}$ is described relative to $\mathcal{F}$ by the variables x_{MS} and y_{MS}. θ_{MS} is the relative angle between the X axes of the frames $\mathcal{F}$ and $\mathcal{F}_{MS}$. A sketch of the material table system is shown in Figure 3.

2.2.3 Assembly Table System

The assembly table has slots to store the intermediate and final assemblies. The center of mass of AS is at C_{AS*}. A coordinate frame $\mathcal{F}_{AS}$ is fixed at the center of mass C_{AS*} with axes parallel to the edges of the assembly table. $\mathcal{F}_{AS}$ is described relative to $\mathcal{F}$ by 3 coordinate variables x_{AS}, y_{AS}, and θ_{AS}. A sketch of the assembly table system is shown in Figure 4.

2.3 Kinematics

2.3.1 Robot System during Free Motion

With the assumption that the center of mass of RS is at C_{RS*}, the 11 variables must satisfy 2 constraints:

$$m_0\mathbf{r}_{0*} + m_1^A\mathbf{r}_{1*}^A + m_2^A\mathbf{r}_{2*}^A + m_3^A\mathbf{r}_{3*}^A + m_1^B\mathbf{r}_{1*}^B + m_2^B\mathbf{r}_{2*}^B + m_3^B\mathbf{r}_{3*}^B = m_{RS}\mathbf{r}_{RS*} \quad (1)$$

where the position vectors are to the center of mass of the respective links in $\mathcal{F}$. On time differentiating the above equation and collecting the terms, it can be written in the following form:

$$a_{11}\dot{x}_{0,RS} + a_{13}\dot{\theta}_0 + a_{14}\dot{\theta}_1^A + a_{15}\dot{\theta}_2^A + a_{16}\dot{\theta}_3^A + a_{17}\dot{\theta}_1^B + a_{18}\dot{\theta}_2^B + a_{19}\dot{\theta}_3^B = 0 \quad (2)$$

$$a_{22}\dot{y}_{0,RS} + a_{23}\dot{\theta}_0 + a_{24}\dot{\theta}_1^A + a_{25}\dot{\theta}_2^A + a_{26}\dot{\theta}_3^A + a_{27}\dot{\theta}_1^B + a_{28}\dot{\theta}_2^B + a_{29}\dot{\theta}_3^B = 0 \quad (3)$$

where the coefficients a_{ij} are functions of geometry and inertial parameters of RS.

During free motion, the applied joint actuator torques are internal. As a result, the linear momentum of RS in the plane and angular momentum normal to the plane remain constant. These three equations can be written as:

$$m_{RS}\dot{x}_{RS} = K_1 \quad (4)$$

$$m_{RS}\dot{y}_{RS} = K_2 \quad (5)$$

$$a_{33}\dot{\theta}_0 + a_{34}\dot{\theta}_1^A + a_{35}\dot{\theta}_2^A + a_{36}\dot{\theta}_3^A + a_{37}\dot{\theta}_1^B + a_{38}\dot{\theta}_2^B + a_{39}\dot{\theta}_3^B = K_3 \quad (6)$$

where m_{RS} is the mass of the robot system and K_1, K_2, K_3 are constant values of momentum components during free motion. These equations do not hold when the robot is acted on by external forces during propulsion, docking, and collision.

2.3.2 Robot System with Locked Joints

With the six joints locked, RS becomes a single rigid body. Hence, $x_{0,RS}$, $y_{0,RS}$ become dependent on x_{RS}, y_{RS}, and θ_0. Hence, it is more convenient to describe the robot system by 3 independent variables: x_{RS}, y_{RS}, and θ_0. In order to facilitate the developments of this paper, we define a vector $\mathbf{X}_{RS} = (x_{RS}, y_{RS}, \theta_0)^T$. Unless acted on by external forces or impacts, $\dot{\mathbf{X}}_{RS}$ remains constant.

2.4 Table systems

We define the vector $\mathbf{X}_{MS} = (x_{MS}, y_{MS}, \theta_{MS})^T$ to describe the motion of the material system and $\mathbf{X}_{AS} = (x_{AS}, y_{AS}, \theta_{AS})^T$ to describe the motion of the assembly table system. The rates $\dot{\mathbf{X}}_{MS}$ remain constant during motion unless MS is acted on by external forces or there is collision. Similarly, $\dot{\mathbf{X}}_{AS}$ remain constant during motion when the assembly table is not acted on by external forces.

3 Models of Assembly Primitives

In this paper, we will address the following assembly primitives: (i) Self motion of the robot system, (ii) Propulsion of the robot system, (iii) Docking of the robot system, (iv) Pickup of a workpiece by the robot system, and (v) Release of a workpiece by the robot system. As mentioned earlier, the joints of the robot are locked during propulsion, docking, pickup, and release and are actively coordinated during self motion.

3.1 Self Motion of the Robot System

During self motion of the robot, with prescribed motion of the six joint angles, the time histories of base coordinates $x_{0,RS}$ and $y_{0,RS}$ are computed using Eqs. (2), (3). The position of the center of mass is governed by (4) and (5) and the orientation angle θ_0 is computed using (6). The center of mass of RS drifts with a constant velocity during self motion.

3.2 Propulsion of the Robot System

The robot is propelled by 2 air thrusters placed at T_1 and T_2 on the base of the robot. The rates of RS during propulsion satisfy the following relation:

$$M_{RS}\ddot{\mathbf{X}}_{RS} = Jv(T_1)^T\mathbf{F}(T_1) + Jv(T_2)^T\mathbf{F}(T_2) \quad (7)$$

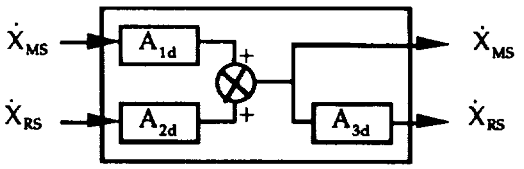


Figure 5: A block diagram of the rate relations for the dock primitive.

where M_{RS} is the inertia matrix of the robot system with respect to $\mathbf{X}_{RS}$, $Jv(T_1)$ and $Jv(T_2)$ are respectively the velocity Jacobians for the thruster locations T_1 and T_2 with respect to $\dot{\mathbf{X}}_{RS}$. $\mathbf{F}(T_1)$ and $\mathbf{F}(T_2)$ are (2×1) thrust vectors described in $\mathcal{F}$.

Given $\mathbf{X}_{RS}$, $\dot{\mathbf{X}}_{RS}$ at initial and final positions, time histories of the thruster forces $\mathbf{F}(T_1)$ and $\mathbf{F}(T_2)$ can be selected in a number of ways to satisfy the conditions at the two end points. A relatively simple way to achieve this is by selecting cubic trajectories for $\mathbf{X}_{RS}$ components that satisfy the end conditions. $\ddot{\mathbf{X}}_{RS}$ computed from these cubic trajectories can then be used to determine the thrust vectors as functions of time.

3.3 Docking of the Robot System

Assume that RS docks with MS such that after docking a point P of RS acquires the same velocity as P' of MS and the two systems after docking acquire the same angular velocity. The analytical model of this primitive is based on collision theory between two rigid bodies [8]. The equations of impact for RS can be written as:

$$M_{RS}(\dot{\mathbf{X}}_{RS}|_{t+} - \dot{\mathbf{X}}_{RS}|_{t-}) = -J_P^T \int_{t-}^{t+} \mathbf{F}_d dt \quad (8)$$

where J_P is the Jacobian matrix of P on RS, and $\mathbf{F}_d$ is the collision vector $(F_{dx}, F_{dy}, M_{dz})^T$ expressed in $\mathcal{F}$. $t+$ and $t-$ are respectively the time instances after and before collision. A similar equation for MS is:

$$M_{MS}(\dot{\mathbf{X}}_{MS}|_{t+} - \dot{\mathbf{X}}_{MS}|_{t-}) = J_{P'}^T \int_{t-}^{t+} \mathbf{F}_d dt \quad (9)$$

where M_{MS} is the inertia matrix of MS for $\mathbf{X}_{MS}$ and $J_{P'}$ is the Jacobian matrix of point P' on MS. After impact, $\dot{\mathbf{X}}_{RS}$ and $\dot{\mathbf{X}}_{MS}$ are related as follows:

$$J_P\dot{\mathbf{X}}_{RS}|_{t+} = J_{P'}\dot{\mathbf{X}}_{MS}|_{t+} \quad (10)$$

On simultaneously solving these three equations, we obtain:

$$\dot{\mathbf{X}}_{MS}|_{t+} = [M_{MS} + J_{P'}{}^T J_P{}^{-T} M_{RS} J_P{}^{-1} J_{P'}]^{-1} \\ [M_{MS}\dot{\mathbf{X}}_{MS}|_{t-} + J_{P'}{}^T J_P{}^{-T} M_{RS}\dot{\mathbf{X}}_{RS}|_{t-}] \quad (11)$$

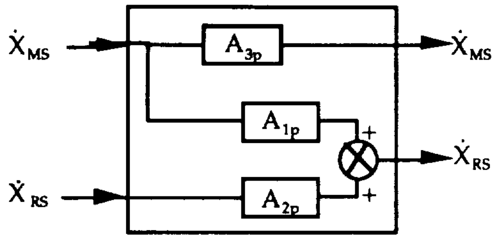


Figure 6: A block diagram of the rate relations for the 'pick' primitive.

and

$$\dot{\mathbf{X}}_{RS}|_{t+} = J_P{}^{-1}J_{P'}\dot{\mathbf{X}}_{MS}|_{t+} \qquad (12)$$

In order to concisely write the above two equations, we define the following matrices:

$$\begin{aligned} A_{1d} &= [M_{MS} + J_{P'}{}^T J_P{}^{-T} M_{RS} J_P{}^{-1} J_{P'}]^{-1} M_{MS} \\ A_{2d} &= [M_{MS} + J_{P'}{}^T J_P{}^{-T} M_{RS} J_P{}^{-1} J_{P'}]^{-1} J_{P'}{}^T \\ &\quad J_P{}^{-T} M_{RS} \\ A_{3d} &= J_P{}^{-1} J_{P'} \end{aligned} \qquad (13)$$

The rates of the two sytem can then be written as:

$$\begin{aligned} \dot{\mathbf{X}}_{MS}|_{t+} &= A_{1d}\dot{\mathbf{X}}_{MS}|_{t-} + A_{2d}\dot{\mathbf{X}}_{RS}|_{t-} \\ \dot{\mathbf{X}}_{RS}|_{t+} &= A_{3d}\dot{\mathbf{X}}_{MS}|_{t+} \end{aligned} \qquad (14)$$

A block diagram of the docking primitive is shown in Fig. 5. It must be noted that M_{RS} and M_{MS} depend on the definitions of the two systems at $t-$, $J_{P'}$ also depends on location of P' on MS, and J_P depends on joint angles of RS.

3.4 Object Pickup by the Robot System

Once RS has docked with MS and is ready to pickup W_i, this primitive relates the rates of RS and MS before and after pickup. It is assumed that during pickup the applied forces are normal to the plane of motion. The changes in the rates, therefore, occur due to redefinition of the two systems RS and MS. In the new definition, W_i is added to RS and W_i has been taken away from MS.

As a result of adding W_i to RS, it has a new position and velocity of the center of mass. The position of the new center of mass of RS is computed from the positions of C_{RS*} and C_{Wi*}.

$$\begin{aligned} x_{RS}|_{t+} &= \frac{m_{RS}|_{t-}x_{RS}|_{t-} + m_{Wi}x_{Wi*}|_{t-}}{m_{RS}|_{t-} + m_{Wi}} \\ y_{RS}|_{t+} &= \frac{m_{RS}|_{t-}y_{RS}|_{t-} + m_{Wi}y_{Wi*}|_{t-}}{m_{RS}|_{t-} + m_{Wi}} \end{aligned} \qquad (15)$$

The velocity of the new center of mass is computed from the velocities of C_{RS*} and C_{Wi*}.

$$\begin{aligned} \dot{x}_{RS}|_{t+} &= \frac{m_{RS}|_{t-}\dot{x}_{RS}|_{t-} + m_{Wi}\dot{x}_{Wi*}|_{t-}}{m_{RS}|_{t-} + m_{Wi}} \\ \dot{y}_{RS}|_{t+} &= \frac{m_{RS}|_{t-}\dot{y}_{RS}|_{t-} + m_{Wi}\dot{y}_{Wi*}|_{t-}}{m_{RS}|_{t-} + m_{Wi}} \end{aligned} \qquad (16)$$

The angular rate does not change as a result of pickup becuase the acting forces are normal to the plane of motion. Hence, $\dot{\theta}_{RS}|_{t+} = \dot{\theta}_{RS}|_{t-}$. Using the velocity Jacobian of C_{Wi*}, the rates before and after pickup can be related as:

$$\dot{\mathbf{X}}_{RS}|_{t+} = A_{1p}\dot{\mathbf{X}}_{RS}|_{t-} + A_{2p}\dot{\mathbf{X}}_{MS}|_{t-} \qquad (17)$$

where A_{1p}, A_{2p} are defined as:

$$\begin{aligned} A_{1p} &= \begin{bmatrix} \frac{m_{RS}|_{t-}}{m_{RS}|_{t-}+m_{W_i}} & 0 & 0 \\ 0 & \frac{m_{RS}|_{t-}}{m_{RS}|_{t-}+m_{W_i}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ A_{2p} &= \frac{m_{W_i}}{m_{RS}|_{t-} + m_{W_i}} \begin{bmatrix} 1 & 0 & y_{RS*Wi*} \\ 0 & 1 & -x_{RS*Wi*} \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \qquad (18)$$

and (x_{RS*Wi*}, y_{RS*Wi*}) are X and Y components of the vector from C_{RS*} to C_{Wi*} expressed in $\mathcal{F}$.

As a result of losing W_i, MS has a new position and velocity of the center of mass. The new center of mass is computed from the positions of C_{MS*} and C_{Wi*}.

$$\begin{aligned} x_{MS}|_{t+} &= \frac{m_{MS}|_{t-}x_{MS}|_{t-} - m_{Wi}x_{Wi*}|_{t-}}{m_{MS}|_{t-} - m_{Wi}} \\ y_{MS}|_{t+} &= \frac{m_{MS}|_{t-}y_{MS}|_{t-} + m_{Wi}y_{Wi*}|_{t-}}{m_{MS}|_{t-} - m_{Wi}} \end{aligned} \qquad (19)$$

Similarly, velocity of new C_{MS*} is computed from the velocities of the old C_{MS*} and W_i.

$$\begin{aligned} \dot{x}_{MS}|_{t+} &= \frac{m_{MS}|_{t-}\dot{x}_{MS}|_{t-} - m_{Wi}\dot{x}_{Wi*}|_{t-}}{m_{MS}|_{t-} - m_{Wi}} \\ \dot{y}_{MS}|_{t+} &= \frac{m_{MS}|_{t-}\dot{y}_{MS}|_{t-} - m_{Wi}\dot{y}_{Wi*}|_{t-}}{m_{MS}|_{t-} - m_{Wi}} \end{aligned} \qquad (20)$$

The two rate relations can be restructured in a matrix form:

$$\dot{\mathbf{X}}_{MS}|_{t+} = A_{3p}\dot{\mathbf{X}}_{MS}|_{t-} \qquad (21)$$

where A_{3p} is defined as:

$$A_{3p} = \begin{bmatrix} k_1 - k_2 & 0 & -k_2 y_{MS*Wi*} \\ 0 & k_1 - k_2 & k_2 x_{MS*Wi*} \\ 0 & 0 & 0 \end{bmatrix} \qquad (22)$$

where $k_1 = \frac{m_{MS}|_{t-}}{m_{MS}|_{t-} - m_{W_i}}$, $k_2 = \frac{m_{W_i}}{m_{MS}|_{t-} - m_{W_i}}$, and (x_{RM*Wi*}, y_{RM*Wi*}) are X and Y components

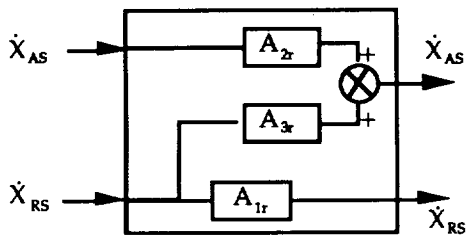


Figure 7: A block diagram of the rate relations for the 'release' primitive.

of the vector from C_{MS*} to C_{Wi*} expressed in $\mathcal{F}$. A block diagram of this primitive is shown in Figure 6. From this block diagram, we can notice that out of the three vectors $\dot{\mathbf{X}}_{MS}|_{t+}$, $\dot{\mathbf{X}}_{MS}|_{t-}$, and $\dot{\mathbf{X}}_{RS}|_{t-}$, any two can be chosen independently. For example, if $\dot{\mathbf{X}}_{MS}|_{t+}$ and $\dot{\mathbf{X}}_{MS}|_{t-}$ are specified $\dot{\mathbf{X}}_{RS}|_{t-}$ and $\dot{\mathbf{X}}_{RS}|_{t+}$ can be uniquely determined.

3.5 Release of an Object by the Robot System

This primitive relates the rates of RS and AS once RS releases an object W_i on AS. It is assumed that during release the applied forces are normal to the plane of motion. The changes in the rates, therefore, occur only due to redefinition of the systems.

As a result of removing W_i from RS, it has a new position and velocity of the center of mass. The position of the new center of mass of RS is computed from the positions of C_{RS*} and C_{Wi*}.

$$x_{RS}|_{t+} = \frac{m_{RS}|_{t-} x_{RS}|_{t-} - m_{Wi} x_{Wi}|_{t-}}{m_{RS}|_{t-} - m_{Wi}}$$
$$y_{RS}|_{t+} = \frac{m_{RS}|_{t-} y_{RS}|_{t-} - m_{Wi} y_{Wi}|_{t-}}{m_{RS}|_{t-} - m_{Wi}} \quad (23)$$

Similarly, the velocity of the new center of mass is:

$$\dot{x}_{RS}|_{t+} = \frac{m_{RS}|_{t-} \dot{x}_{RS}|_{t-} - m_{Wi} \dot{x}_{Wi}|_{t-}}{m_{RS}|_{t-} - m_{Wi}}$$
$$\dot{y}_{RS}|_{t+} = \frac{m_{RS}|_{t-} \dot{y}_{RS}|_{t-} - m_{Wi} \dot{y}_{Wi}|_{t-}}{m_{RS}|_{t-} - m_{Wi}} \quad (24)$$

The two rate relations can be restructured in a matrix form:

$$\dot{\mathbf{X}}_{RS}|_{t+} = A_{1r} \dot{\mathbf{X}}_{RS}|_{t-} \quad (25)$$

where A_{1r} is defined as:

$$A_{3p} = \begin{bmatrix} k_3 - k_4 & 0 & -k_4 y_{RS*Wi*} \\ 0 & k_3 - k_4 & k_4 x_{RS*Wi*} \\ 0 & 0 & 0 \end{bmatrix} \quad (26)$$

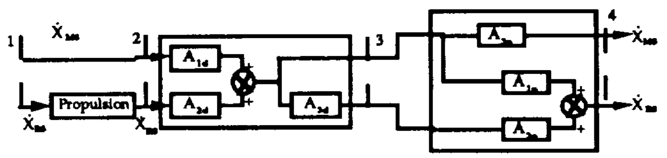


Figure 8: A block diagram of the 'propel/dock/pick' primitive.

where $k_3 = \frac{m_{RS}|_{t-}}{m_{RS}|_{t-} - m_{W_i}}$, $k_4 = \frac{m_{Wi}}{m_{RS}|_{t-} - m_{W_i}}$, and (x_{RS*Wi*}, y_{RS*Wi*}) are X and Y components of the vector from C_{RS*} to C_{Wi*} expressed in $\mathcal{F}$.

The velocity of the new center of mass of AS is computed from the velocities of C_{AS*} and C_{Wi*}.

$$x_{AS}|_{t+} = \frac{m_{AS}|_{t-} x_{AS}|_{t-} + m_{Wi} x_{Wi}|_{t-}}{m_{AS}|_{t-} + m_{Wi}}$$
$$y_{AS}|_{t+} = \frac{m_{AS}|_{t-} y_{MS}|_{t-} + m_{Wi} y_{Wi}|_{t-}}{m_{AS}|_{t-} + m_{Wi}} \quad (27)$$

Similarly, the new velocity of the center of mass is:

$$\dot{x}_{AS}|_{t+} = \frac{m_{AS}|_{t-} \dot{x}_{AS}|_{t-} + m_{Wi} \dot{x}_{Wi}|_{t-}}{m_{AS}|_{t-} + m_{Wi}}$$
$$\dot{y}_{AS}|_{t+} = \frac{M_{AS}|_{t-} \dot{y}_{AS}|_{t-} + m_{Wi} \dot{y}_{Wi}|_{t-}}{m_{AS}|_{t-} + m_{Wi}} \quad (28)$$

The angular rate does not change as a result of adding W_i, hence, $\dot{\theta}_{AS}|_{t+} = \dot{\theta}_{AS}|_{t-}$. Using the velocity Jacobian of C_{Wi*}, the rates before and after pickup can be related as:

$$\dot{\mathbf{X}}_{AS}|_{t+} = A_{2r} \dot{\mathbf{X}}_{AS}|_{t-} + A_{3r} \dot{\mathbf{X}}_{RS}|_{t-} \quad (29)$$

where A_{2r}, A_{3r} are defined as:

$$A_{2r} = \begin{bmatrix} \frac{m_{AS}|_{t-}}{m_{AS}|_{t-} + m_{W_i}} & 0 & 0 \\ 0 & \frac{m_{AS}|_{t-}}{m_{AS}|_{t-} + m_{W_i}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$A_{3r} = \frac{m_{W_i}}{m_{AS}|_{t-} + m_{W_i}} \begin{bmatrix} 1 & 0 & y_{AS*Wi*} \\ 0 & 1 & -x_{AS*Wi*} \\ 0 & 0 & 0 \end{bmatrix} \quad (30)$$

and (x_{AS*Wi*}, y_{AS*Wi*}) are X and Y components of the vector from C_{AS*} to C_{Wi*} expressed in $\mathcal{F}$. A block diagram of this primitive is shown in Figure 7.

4 Modeling of Assembly

4.1 A Simple Assembly

Consider a simple assembly task that requires the robot to pick W_1 and W_2 from MS, assemble these in the form of an 'L' shape, and place this composite body on AS. A possible sequence of primitives

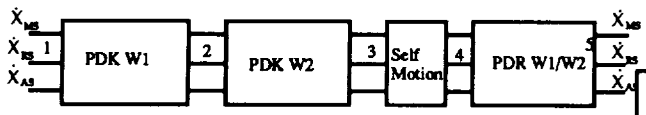


Figure 9: A block diagram of the simple assembly described in Section 4.

to complete this assembly task is: (i) RS propels to W_1, (ii) RS docks with MS to grip W_1 by arm A, (iii) RS picks W_1 from MS, (iv) RS propels to W_2, (v) RS docks with MS to grip W_2 by arm B, (vi) RS picks W_2 from MS, (vii) RS executes self motion to assemble W_1 and W_2, (viii) RS propels to AS, (ix) RS docks with AS to release W_1/W_2, (x) RS releases W_1/W_2 on AS. In this small assembly task, we saw the sequence of primitives propel, dock, and pickup (PDK) repeated twice and the sequence propel, dock, and release (PDR) once. These two sequences of primitives appear quite commonly during assembly and require further study to determine their characteristics.

4.2 Propulsion/dock/pickup (PDK) Sequence

Fig. 8 shows a block diagram of this sequence of primitives. It can be infered from this block diagram that if $\dot{\mathbf{X}}_{MS}$ at nodes 1 and 4 are specified, $\mathbf{X}_{MS}$ at nodes 2, 3 and $\dot{\mathbf{X}}_{RS}$ at nodes 2, 3, 4 are uniquely determined. Also, with the propulsion primitive, for any given $\mathbf{X}_{RS}$ and $\dot{\mathbf{X}}_{RS}$ at node 1, a desired $\mathbf{X}_{RS}$, $\dot{\mathbf{X}}_{RS}$ at node 2 can be reached by suitably selecting a time history of the thruster forces. From these two observations, one can form a broader conclusion that it is possible to achieve any desired $\dot{\mathbf{X}}_{MS}$ at the end of a PDK sequence for arbitrary $\dot{\mathbf{X}}_{MS}$ and $\dot{\mathbf{X}}_{RS}$ at the beginning of the sequence. A similar conclusion can be made for propulsion/dock/release (PDR) sequence. These two conclusions play important roles in developing strategies for assembly.

5 Description of FLOAT

A general purpose program FLOAT was developed to study and test a variety of assembly strategies in a free-floating planar work environment. The inputs to this program consist of (i) the inertial description of the units, (ii) the geometric description of the units, (iii) the assembly sequence in the form of P/D/K/R/S/PDK/PDR commands and strategy of assembly in terms of desired values of $\dot{\mathbf{X}}_{RS}$, $\dot{\mathbf{X}}_{MS}$, $\dot{\mathbf{X}}_{AS}$ at different points of the assembly sequence. The program creates the current RS, MS, and AS while executing a specific primitive. Using the assembly strategy, the program computes the motion plans for RS, MS and AS and updates the coordinate and rate variables of the units. A flowchart for the program for a sequence of three commands P, DK, and S is shown in Fig. 10.

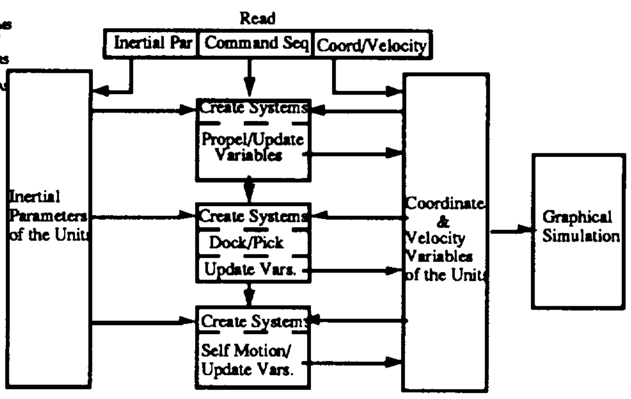


Figure 10: A flowchart of execution of the three commands, P, DK, S using 'FLOAT'.

6 Conclusions

In this paper, we presented a method for analytical modeling of assembly using a free-floating planar robot in a free-floating planar work environment. The model of the assembly was obtained by combining analytical models of five primitives: (i) self motion of the robot, (ii) propulsion of the robot, (iii) docking of the robot, (iv) pickup by the robot, and (v) release by the robot. It was concluded that assemblies typically consist of a number of propulsion, dock, pickup/release sequences interluded by self motion. On examining a PDK sequence, it was observed that starting out from arbitrary velocities of the robot system and material system, it was possible to achieve any desired material system velocities by suitably controlling the thruster forces of the robot system during propulsion. A similar conclusion could be arrived at for a PDR sequence. These observations provide guidelines to select proper velocities of RS, MS, AS at intermediate steps during assembly. A general purpose program was developed to study and test assembly strategies for a variety of assemblies. Even though this paper deals specifically for planar free-floating robots, the concepts can be extended to free-floating spatial robots working in zero gravity environment.

7 Acknowledgments

The support of the Dynamic Systems and Control Division of the National Science Foundation under the Research Initiation Program, 1804 Fund Awards, Baker Fund Awards from Ohio University are gratefully acknowledged.

8 Appendix

8.1 Jacobian for Points on a Rigid Body

Consider a rigid body B undergoing planar motion. The position of a point $B*$ on this body is described in an inertial frame $\mathcal{F}$ by the coordinates x_{B*} and y_{B*}. The orientation of a line $B * B_1$ is described by the angle θ_B. The velocity Jacobian for point B_1 with respect to $\dot{x}_{B*}$, $\dot{y}_{B*}$, $\dot{\theta}_B$ in $\mathcal{F}$ is given as:

$$\begin{bmatrix} V_{B1x} \\ V_{B1y} \\ \dot{\theta}_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & y_{B*B1} \\ 0 & 1 & -x_{B*B1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{B*} \\ \dot{y}_{B*} \\ \dot{\theta}_B \end{bmatrix} \qquad (31)$$

where $(x_{B*B1} and y_{B*B1})^T$ are X and Y components of the vector $\mathbf{r}_{B*B1}$ expressed in $\mathcal{F}$. The (3 x 3) matrix is the Jacobian map for point B_1 labeled as J_{B_1}. The upper (2 x 3) block is the velocity Jacobian $Jv(B1)$.

8.2 Inertia Matrix for a Rigid Body

In Section 8.1, if $B*$ is the center of mass of B, the inertia matrix of body B relative to the coordinates $(x_{B*}, y_{B*}, \theta_B)^T$ is given as:

$$M_B = \begin{bmatrix} m_B & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & I_B \end{bmatrix} \qquad (32)$$

where m_B is the mass of the body B and I_B is the centroidal moment of inertia about an axis normal to the plane of motion.

References

[1] Agrawal, S.K. and Garimella, R.,"Workspace Boundaries of Free-floating Open and Closed-Chain Planar Manipulators", to appear in *Journal of Mechanical Design, Transactions of the ASME*, 1993.

[2] Agrawal, S.K. and Desmier, G.,"Kinematics of Assembly Primitives for Free-floating Robots", In Proceedings, *IEEE International Conference on Intelligent Robot Systems*, Yokohama, Japan, 1993.

[3] Agrawal, S.K. and Garimella, R.,"Path-planning of a Free-floating Closed-Chain Planar Manipulator Using Inverse Kinematics", under review, *Journal of Mechanical Design, Transactions of the ASME*, 1993.

[4] Agrawal, S.K. and Shirumalla, S.,"Motion planning of a Dual-arm Free-floating Planar Manipulator with Inertially Fixed Base", to appear in *Mechanism and Machine Theory*, 1993.

[5] Agrawal, S.K. and Krajnak, J.A.,"Design of a Quadrilateral Pneumatic Thruster for a Free-floating Robot", under review *IEEE Transactions on Robotics and Automation*, 1993.

[6] Agrawal, S.K., Rambhaskar, J., and Annapragada, M.,"Design of a New Contactless Sensor for Robotic Applications", under review *IEEE Transactions on Robotics and Automation*, 1993.

[7] Alexander, H.L. and Cannon, R.H.,"An Extended Operational Space Control Algorithm for Satellite Manipulators", *The Journal of Astronautical Sciences*, vol. 38, no. 4, 1990, pp. 473-486.

[8] Kane, T.R. and Levinson, D.A., *Dynamics: Theory and Applications*, McGraw-Hill Book Company, New York, 1985.

[9] Nakamura, Y. and Mukherjee, R., "Nonholonomic Path Planning of Space Robots Via Biirectional Approach", *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, 1991, pp. 500-514.

[10] Papadopoulos, E. and Dubowsky, S., "On the Dynamic Singularities in the Control of Space Manipulators", *ASME Winter Annual Meeting*, 1989, pp. 45-51.

[11] Schneider, S.A. and Cannon, R.H.,"Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results", *IEEE Transactions on Robotics and Automation*, vol. 8, June 1992, pp. 383-394.

[12] Vafa, Z., "Space Manipulator with No Satellite Attitude Disturbance", *IEEE International Conference on Robotics and Automation*, 1990, pp. 1770-1775.

[13] Yoshida, K., Kurazume, R., and Umetani, Y., "Dual Arm Coordination in Space Free-flying Robot", In proceedings, *IEEE International Conference on Robotics and Automation*, vol.3, April 1991, pp. 2516-2521.

Tuesday
March 22, 1994

C-3

6042

AIAA-94-1196-CP

DESIGN AND CONTROL OF ACTIVE VISION BASED MECHANISMS FOR INTELLIGENT ROBOTS

N94- 30547

S31-63
207644
P-10

Liwei Wu
email: liweiwu@ece.arizona.edu

Michael M. Marefat
email: marefat@ece.arizona.edu

Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721

ABSTRACT

In this paper, we propose a design of an active vision system for intelligent robot application purposes. The system has the degrees of freedom of pan, tilt, vergence, camera height adjustment and baseline adjustment with a hierarchical control system structure. Based on this vision system, we discuss two problems involved in the binocular gaze stabilization process. They are fixation point selection, vergence disparity extraction A hierarchical approach to determining point of fixation from potential gaze targets using evaluation function representing human visual behavior to outside stimuli is suggested. We also characterize different visual tasks in two cameras for vergence control purposes and phase-based method based on binarized images to extract vergence disparity for vergence control is presented. Control algorithm for vergence control is discussed.

I. Introduction

The advantages of active vision over passive vision in enabling the robot to explore its environment and then to adapt to the environment have been recognized by many researchers in active vision paradigm. As defined by Ruzena Bajcsy [1], active vision is a problem of intelligent control applied to data acquisition process depending on the goal or task of the process. It is able for the active vision system to improve its view point to overcome the inherent problem involved in passive vision that the sensor only takes in those percepts that randomly fall onto the sensors and thus, enlarges active vision based robot's adaptability to its environment.

From this definition we can elicit two points. The first is *what we want to see* (data acquisition depending on the goal or task of the process.). This is the problem of visual target selection. The second idea is *how to see the selected target* (intelligent control applied to data acquisition.). This involves determination of the position of the target and control of the vision system such that the target can be percepted. See Fig 1.1.

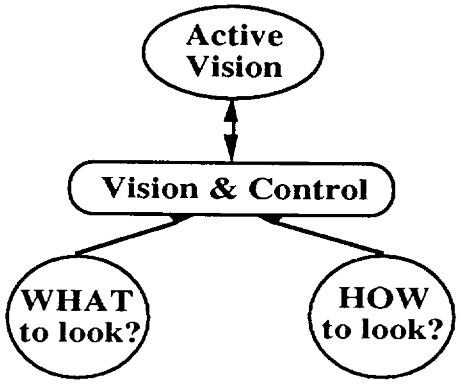


Fig 1.1 Concepts of an active vision system

Of importance to active vision is the gaze control strategy. Gaze control can be roughly partitioned into two categories [2]: Gaze Stabilization, which consists of controlling the available degrees of freedom for the active vision system such that clear images of interesting world point is maintained, and Gaze Change, which is motivated by the need to reduce computational complexity of visual tasks or to gaze at a new point that is taken into account for the visual tasks. This paper is concerned with problems in gaze stabilization.

From the point of view of binocular visual system, gaze stabilization means the visual axis of the two cameras point at the point of interest. The process of gazing at such a point is referred to as *fixating* and the point to be fixated at is known as *point of fixation*. Holding gaze at a selected target has several advantages in image processing. Gazing at the selected target means to capture the target in the part of the lens with highest resolution. This helps quantitative or qualitative visual performance. When the target is near the origin of an image, perspective projection model, which involves non-linearity, can be replaced by orthographic projection model that simplifies many computations. Since the fixation point has a stereoscopic disparity of zero, it is possible to use stereo algorithm that accepts limited range of disparity. This undoubtedly accelerates image processing. While the target is moving, fixating at it induces target "pop-out" [5] due to motion blur so that segmentation is much easier.

Basicly there are three problems involved in gaze stabilization, see Fig 1.2.

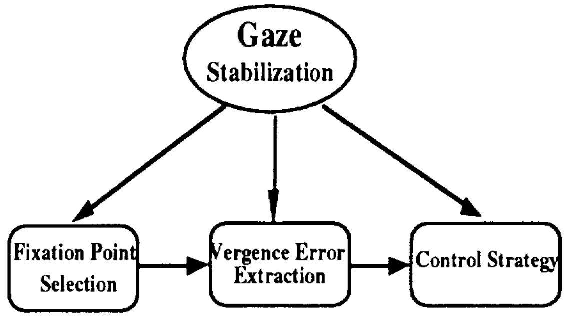


Fig 1.2 Three problems involved in gaze stabilization

The first problem in gaze stabilization is the determination of point of fixation FP. It is the first step in gaze stabilization. Gazing without a fixation point is ridiculous. The determination or selection of a point of fixation is to find the image coordinates of the fixation point's projection in the image plane in the presence of many alternatives based on some criteria. As active vision is a purposeful perception of visual targets, the selection of fixation point will depend on the goal of visual tasks.

The second problem is vergence disparity measurement. The process of two visual sensors' pan motion about their vertical axes in opposite direction to fixate at the selected point of fixation is called *vergence*. Since the optical axes are initially not pointing at a selected point of fixation, the vergence error must be derived so that they can be compensated for to ensure that both optical axis are keeping directed at the target.

The third problem is also the key point of general active vision research. An active vision system has mechanisms that can actively control camera parameters such as position, orientation, vergence, focus, aperture, etc. in response to the requirements of the task. Active vision system is, thus, not only a visual system but also a control system. The tasks of an active vision system are not only visual tasks but also control tasks. Therefore the third problem is the control strategy by which gaze stabilization can be fulfilled.

In this paper we are going to present the design of an active vision system and deal with these problems in binocular system's gaze stabilization with emphasis on fixation point selection and vergence disparity extraction. We introduce the concept of fixation point candidates (FPC's) in the image the cameras take and use evaluation functions to hierarchically determine the point of fixation among all the candidates. This approach is a mathematical representation of psychological results of human visual behavior so that our approach has a solid theoretical foundation. Based on binarized images, we propose a method that robustly and efficiently extract vergence disparity signal, i.e., the vergence error. This error is the motivation of corresponding vergence control action of binocular system to ensure gaze stabilization. The method has certain advantages over existing approaches discussed in [3] and [5].

The paper is organized as follows. In the coming section, the design of our robot "head", i.e., the binocular active vision system will be presented followed in section III by the discussion of the approach to determining point of fixation, Then in section IV, vergence disparity extraction is discussed. The paper ends with conclusion in section VI.

II. A Binocular Active Vision System

1. Robot "Head"

To implement binocular active gaze stabilization, a particular apparatus is required to provide control over the acquisition of image data. From a mechanical perspective, a binocular active system has a mechanical structure which provides mechanisms for modifying the geometric or optical properties of two cameras mounted on it under computer control. One approach is the construction of a robot "head". The design of such a robot "head" includes the design of a mechanical structure on which the cameras are mounted, by which cameras positioning can be completed as well as the design of a control system that controls the cameras' movement and also camera's optical parameters (which is not going to be discussed in this paper.).

A robot "head" has at least the following degrees of freedom:
1) Pan, which is a rotation of the two cameras about a vertical axis passing the midpoint of the baseline;
2) Tilt, which is a rotation of the two cameras about a horizontal axis, e.g., the baseline;
3) Vergence, which is an antisymmetric rotation of each camera about a vertical axes passing through each camera.. See Fig 2.1 and Fig 2.2.

Several research groups have built some robotic heads subject to different design criteria and applications. As a matter of fact, different realization has its own advantages and disadvantages. As to active vision sensors, what is more important, it seems to us, is the ability to obtain accurate 3-D information and convenience implementation of gaze control. Baseline adjustment ability is added to the system in our "head" design apart from other degrees of freedom. Baseline adjustment is the change of distance between two vertical axes of the two cameras, assuming the vertical axis pass the focal point. It is considered to enhance the ability for accurate depth perception when the vision system is close to the

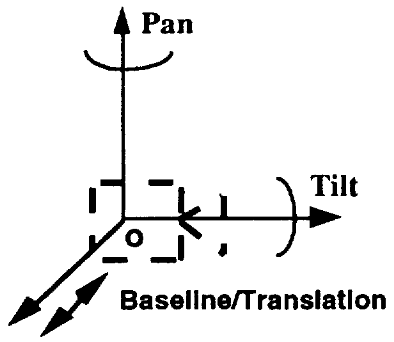


Fig 2.1 Pan, tilt motion of the robot head

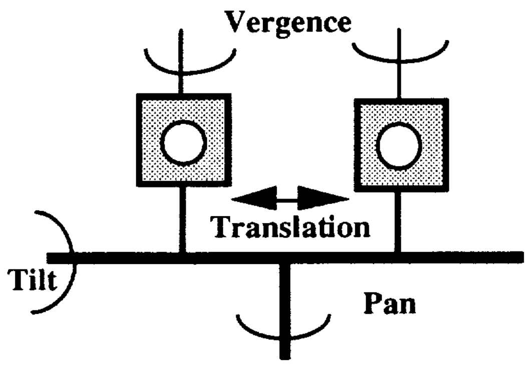


Fig 2.2 Degrees of freedom of the robot "head"

object, although the "baseline" of human visual system is fixed. Thus the cameras can translate along tilt axis. Note, this translation movement is antisymmetric. Secondly, the gaze ability of a binocular active vision system is the most significant advantage over any other types of vision system. We choose the structure as shown above in Fig 2.2 because this structure has several advantages over other possible designs in gaze control. In this design, the vergence angle and pan angle are controlled by separate motors (Pan angle is controlled by pan motor and vergence angle by vergence motors.) and are orthogonal -- either parameter can be altered without disturbing the other [3]. A mechanical advantage of this design is its simplicity: the compact mechanisms and fairly direct linkages facilitate rapid saccades change[3]. The structure of our robot "head" is depicted in Fig 2.3, where head's height adjustment ability is added in case of necessity.

2. "Head" on a Robot Arm

Although the "head" is provided with pan, tilt, vergence, and baseline adjustment motion abilities to change the cameras positioning and orientation to obtain various viewpoint for different tasks, there are still some vision problems in application that such a "head" cannot solve. Active vision system is not merely a vision system, it serves for action. It will cooperate with a robot arm to accomplish a specific task. In real application, the view could be obstructed when the robot arm is in close proximity to the object. Also, in CIM applications, the "head" may need to see the opposite face or a side face of a part. In such cases, we can clearly feel that more "degree of freedom" should be provided to the visual system, the head. This means that it is better to mount the vision head on the end-effector of a robot arm (See Fig 2.3). This configuration will offer maximum field of view for the cameras.

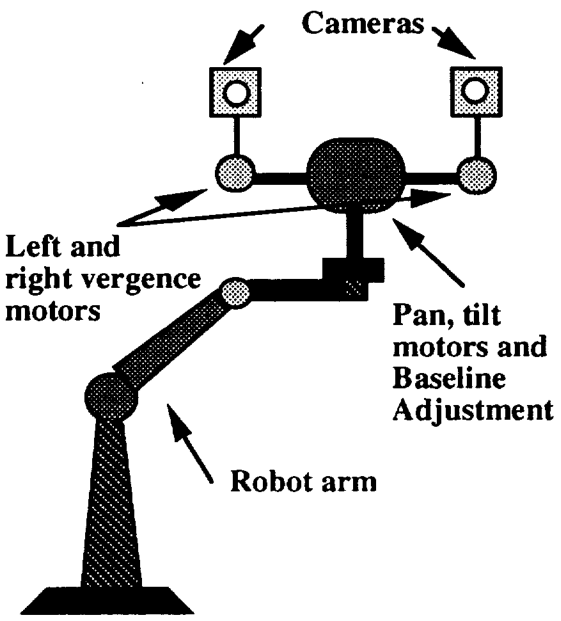

Cameras
Left and right vergence motors
Pan, tilt motors and Baseline Adjustment
Robot arm

Fig 2.3 A "head" mounted on the end-effector of a robot arm

3. Robot Head's Control System Blocks

Each degree of freedom is actuated by a DC servo motor because of its easy controllability nature. The basic block diagram of the robot head's control system is shown in Fig 2.4. Each degree of freedom has its own local controller, which are coordinated by the robot head platform control block. The control block is interfaced to a host computer which is also the host computer of the whole active vision system. Control signals are synthesized in the host computer and sent to platform control block. The control block receives the command from the host, does kinematic calculation to get control signal for pan, tilt, vergence, or other motion control purposes, and then sends them to different local controllers to implement the control command from the host computer. The system forms a hierarchical control structure with three levels. The top level is the host. In the middle, platform sub-controller communicates with host and the bottom level local controllers as a coordinator. The bottom level local controllers are actual controllers for specific control task, such as pan, tilt, or vergence,etc.

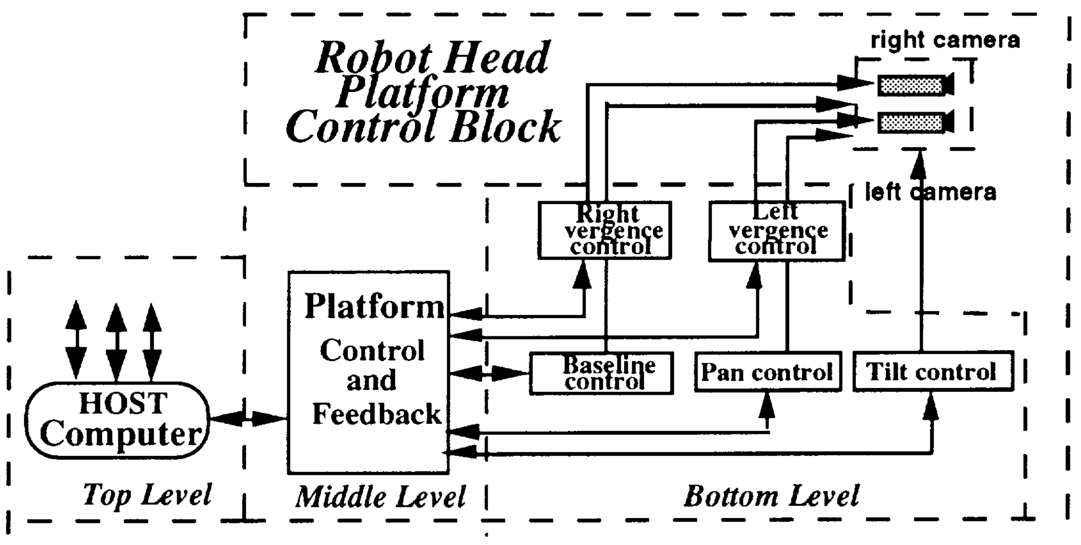


Fig 2.4 Robot head's control system block diagram

III. Determination of Point of Fixation

The general gaze stabilization problem is to maintain fixation on a (moving) visual target from a moving observer. In our case of binocular system, this means the axis of the two cameras point at the target. Thus, the positions of the projections of the target are at the origins of both image plane coordinate frames. Since the object the vision system "looks" is usually not a geometric point that has no volume the projection of the object in the image plane will not be a point but an area. Then the first question we encounter is "what part of the object should the cameras fixate at"?

1. Gaze Target and Its Selection

Gaze stabilization is closely related to visual tasks the system performs. The goal of present visual task determines what the system should gaze. This is true because focusing limited system resources on restricted region of the scene, or the most important region of a scene related to current visual task, is necessary from the point of view of cost and complexity considerations [2]. In this paper, we are not going to discuss the problem of "What I am going to look". This is related to "next look" problem and is beyond the scope of our discussion in this paper. What we discuss is the mechanism of gaze stabilization. The problem is "How I am going to look". This means we will tell the system what it should look. Once it is told what to look, it is system's responsibility to find the target and hold gaze at it.
Some human visual behaviors form our theoretical foundation of selection of gaze target. Human visual shifts when the visual systems are confront with a new stimulus. This stimulus will then become the new target the eyes are to fixate at. The shift is wholly dependent on the visual information and the result of the shift is to bring the target onto the fovea, where resolution is highest. Psychological studies of human visual behavior to outside stimuli reveal that any detectable feature can be used to guide attentional shift, but color, high-contrast region and image area with high spatial frequency being important factors in visual search and that attention often shifts to areas of "information detail". In a simple case, when searching random 2-D polygonal form, eye fixation tends to concentrate on vertices. These two criteria are called Low-level visual stimuli criterion and High-level visual stimuli criterion, respectively [4].

Hence, the targets that the system may hold gaze at are corners/vertices or edge points in an image. We choose them as potential targets not only because of the fact that human visual attention often shifts to areas of "information detail [4] such as vertices, edges, and axis of symmetry, etc. but also, on the other hand, corners/vertices and edge points are the most "salient" features in a picture and are of extremely usefulness in vision research. Finally, corners/vertices and edge points are more "explicit" features than others that can be used for study of gaze stabilization. Generally speaking, we choose the most "salient" and "explicitly represented" feature in an object as our promising fixation target. Our fixation point selection is feature-based.

To select the point of fixation from among all the corners/vertices and edge points in a picture, we need a couple of tools. One is the approach to selecting it from all the regular corners/vertices and edge points. We use a hierarchical approach to find the gaze target, the fixation point. The other is the criterion used to help in the selection of point of fixation from potential candidates. The criterion will be represented in the form of evaluation function. Practically, when we are selecting our gaze target, these two tools are used combinedly. The process of gaze target selection is described in Fig 3.1.

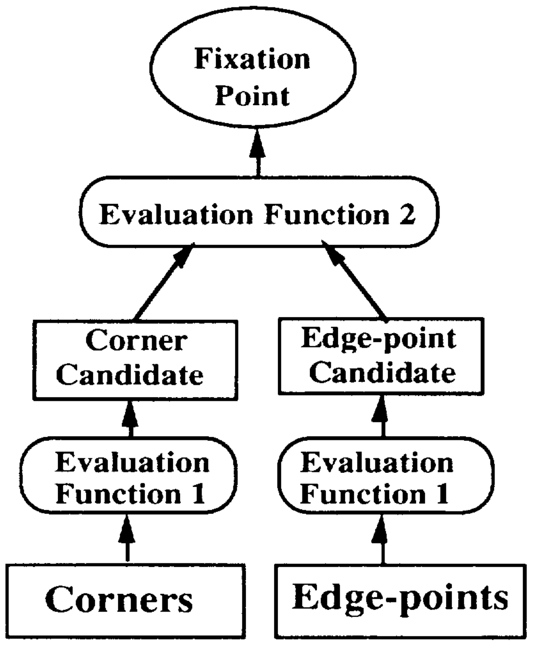


Fig 3.1 A Hierarchical approach to the determination of fixation point

We first find all the corners/vertices and edge points in a picture. They form two separate groups. In each group, we use evaluation function to determine each group's possible gaze target (fixation point), which is called *fixation point candidate*. Between the two candidates, we again apply evaluation function (different from the former evaluation function in parameters, structure, and etc.) to find the gaze target, the fixation point. The detailed algorithm will be given in the later sections. In the following two sub-sections, we will first discuss detection of corners and special edge points in an image which form the mentioned candidate groups.

2. Corners and Special Edge Points

A. Related Work to Corner Detection

Corner detector as an image feature extractor has been discussed in many literature. Corners/vertices are important features of an object. They can be used for identification of an object in the scene, for stereoscopic matching, and displacement vector measuring [6]. In binocular system's gaze stabilization they are considered to be the most important fixation point candidates.

Since corner is also an edge point where curvature changes drastically, in the earlier approaches to detect a corner/vertex, image is first segmented and then the curvature of edges is computed. A corner/vertex is declared if the curvature at the point is greater than a predefined threshold and the point is also an edge point [8]. The other group of approaches of corner/vertex detection i.e., more recent approaches, is based directly on gray-level image. The effort was first made by Beaudet [7]. These methods measure the gradients of the image and use an operator to measure the "cornerness". These methods can be referred to [8][9][10][11], which are considered to be equivalent in nature [11].

An appropriate approach to corner detection for gaze stabilization application can be found in [18]. The approach searches for edges according to the gradient magnitude and direction to find a micro-intersection points, calculation of the distance from the intersection to the current point and keep of the minimum distance. After non-minimum suppression in the distance distribution map, all corners can be found. The algorithm is simple, reliable and noise insensitive and has good localization [18]. These are important reasons that this approach is chosen for our real-time corner-detection application.

B. Special Edge Points

Edge points are another class of "salient" features that can be considered as gaze target in gaze stabilization. Clearly, we are unable to search for edge candidate from among all the edge points since it is computationally much too expensive to do that. And in fact, it is not necessary to consider all the edge points. Physiological research tells us some other interesting properties of human visual behavior to outside stimuli. Proximity of Stimuli [4] states that for several potential targets in the visual field, the one which is closest to the fovea is more likely to be selected as a fixation target and Direction of Stimulus states that upward eye movement is preferred to downward movement. We may conclude that, for two potential new targets, the one that lies above and close to current origin of image frame is more likely to be selected as the next fixation target than the positionally lower and far target.

According to proximity stimuli criterion, we say only one specific edge point on an edge line segment that is closest to current origin of the image plane coordinate needs taking into account. An edge point which is closest to another point p_x (here it should be the origin) that does not lie on that edge line segment is the intersection point (p_e) of this edge line segment and the line which passes p_x and is perpendicular to that edge line segment, i.e., the foot of perpendicular. See Fig 3.2 (a).

In order to determine the edge point candidate, we draw vertical lines to each detected edge line segments from the origin of the image plane coordinate. The intersection points thus determined are of interest and from all these special edge points the edge point candidate will be selected.

But note, there are two cases in which the resulting intersection points will not be taken into account. The first case is that the intersection point is one of the

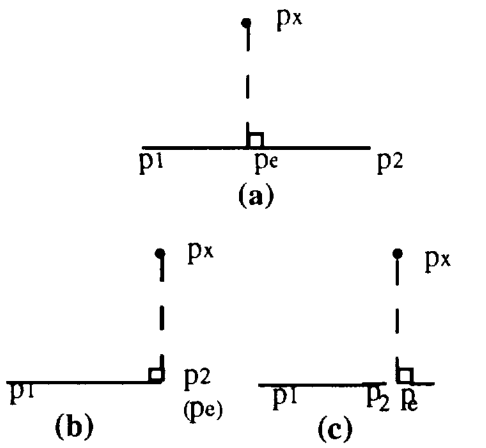


Fig 3.2 (a) Foot of perpendicular. (b) Intersection point is one of the end points. (c) Intersection point lies on the extended line of the edge line segment.

end points of the edge line segment, see Fig 3.2 (b). Since end points are also corners/vertices that have been considered, these intersection points are discarded. The second case is that the intersection point lies on the extended line of the edge line segment , see Fig 3.2 (c). Thus, the computed intersection point actually does not exist. These points also can not be considered. We propose a simple method to detect if a computed intersection point is on the extended line.

In the case of Fig 3.2 (a), point p_e lies on the line segment. we have:

$$\overline{p_1p_e} + \overline{p_ep_2} = \overline{p_1p_2} \quad (3.1)$$

In Fig 3.2 (c) where intersection point lies on the extended line, we have:

$$\overline{p_1p_e} + \overline{p_ep_2} > \overline{p_1p_2} \quad (3.2)$$

When (3.2) holds, we should discard the computed intersection point p_e

C. Fixation Point Candidates Determination

Now, all the corners/vertices detected and edge points that are computed form two groups. We are going to determine the fixation point candidate (FPC's) in each group. The approach to determine the FPC's is based on the psychological studies conclusions on human visual behavior. An evaluation function which represents both proximity of stimulus and direction of stimulus criteria is formulated to aid in the decision making of fixation point candidate selection. This first evaluation function takes the form of:

$$FPC_i = \min\{\alpha X_i^a, X_i^b\} \quad (3.3)$$

where X denotes either a corner (then $X \triangleq C$) or an edge point (then $X \triangleq E$), a and b represent those points that are positionally above or below the current origin of the image plane coordinate frame. X_i (i = 1, 2, ..., j, the number of corners detected or special edge points that are computed.) is computed as Cartesian distance between the point and the origin and thus is:

$$X_i = \sqrt{p_x^2 + p_y^2} \quad (3.4)$$

where p_x and p_y are the coordinate values of the point being considered.

α is a constant between 0 and 1, i.e., $0 < \alpha \leq 1$. This weight represents the criterion of direction of stimulus.

Then the points, a corner and an edge point, will be selected as corner fixation point candidate and edge point fixation point candidate in each group if they have the minimal values of FPC_i in each group. The two selected candidates have the distances C_{FPC} and E_{FPC} from the origin, respectively.

D. Fixation Point Determination

Fixation point will now be determined between the two candidates. The criteria for the selection is also to apply mathematical representation of psychological results in the form of evaluation function. The second evaluation function for the final fixation point selection is:

$$FP = \mathrm{sgn}\{[b^*C_{FPC} - E_{FPC}] + [D(C_{FPC}) - D(E_{FPC})]\} \quad (3.5)$$

where sgn(.) is a sign function and D(·) is the measure of the dimension of the point being considered. If the point lies on one of the coordinate axes, its dimension is 1, otherwise the dimension is 2. This is a measure for control implementation. Larger dimension means more control actions will be concerned.

β is a constant and $0 < \beta \leq 1$. This weight used here represents the intention that corner is more preferred to be selected than edge point candidates due to High-level visual stimuli criterion.

Thus, if FP > 0, which means either the distance and dimension of the corner candidate are greater than those of the edge candidate or much control will be concerned though the distance of the corner candidate is slightly shorter than that of the edge candidate, then the edge point candidate will finally be selected as point of fixation.

If FP < 0, which means the opposite situation to the above discussion, then the corner candidate will finally be selected as point of fixation.

We may derive from the above discussion that the determination of fixation point not only depends on the features themselves but also the weights we select, i.e., α and β. In some sense, the selection of α and β has important influence on decision making on fixation point selection. We propose that $\alpha = 0.9 \sim 0.95$ and $\beta = 0.95 \sim 0.99$.

The algorithm for determination of the point of fixation is given below:

1) For each corner or special edge point in each group, calculate its distance X_i from the local origin using (3.4),
2) Determine the candidate for point of fixation in each group using evaluation function 1 represented by (3.3),
3) Determine the point of fixation using evaluation function 2 represented by (3.5),
4) Get the coordinates of the selected point of fixation: (x_{FPL}, y_{FPL}).

IV. Vergence Disparity Measurement

1. Problem Description

As mentioned before, gaze stabilization in binocular system means pointing the two optical axes of two cameras to the selected fixation point. Thus, the positions of the projection of the fixation point are at the origins of the two image planes. The process of realizing fixation is called *vergence.* A straightforward and easy way to do this is to select the fixation point in different cameras separately and control the parameters of the degrees of freedom available to each camera such that the fixation point projects onto each origin of the image planes coordinate frame. However, this method is not reliable. The reason is that if fixation point is selected separately in two cameras, we are unable to say that the two cameras will select the same point because geometrically the initial positions of projection of the object in two images are quite different. The approach proposed does not guarantee global determination (which means determination of position of a visual target in two images.) of the position of fixation point. This results in non-fixation in real application.

Then , what is a reliable method? Remember the vergence system is also a control system. From the view point of a closed-loop control system, the measure of the difference, or error, between the desired input and the actual output is important since control signal is synthesized based on this error signal [22]. Back to our vergence control, let's ask: "What is the error signal involved in vergence control"? We know that fixation point has a stereoscopic disparity of zero. This is a "salient" feature of fixation. To achieve fixation means to obtain zero disparity between two images. If the disparities between the two cameras are zero, we are sure that the two cameras are fixating at the same point. So to compensate the disparity between two images is a direct and reliable approach to realizing fixation.

If we accept this conclusion and try to find the disparities, one of the images in the two cameras should be considered as the reference image. If the image of the left camera is chosen as reference image, we say the left camera is the dominant camera [4]. Th n, the task of fixation point selection only affects the dominant camera. The tasks involved in the dominant camera and its sub-control system are:

1.(optional) Tracking if the target is in motion with respect to the dominant camera,
2. Fixation point selection, and
3. Control of degrees of freedom to keep the optical axis directed to the fixation point.

Now we can consider the image in the other camera, the non-dominant camera, as the "output" of the vergence system. Then, the difference or the disparity between two images, are the *error* signal of a vergence system. So we need to control the parameters of the degrees of freedom available to the non-dominant camera such that the disparity is compensated. When vergence control results in zero-disparity, we believe that the two cameras fixate at the same target. Therefore, tasks involved in non-dominant camera and its sub-control system are:
1. Vergence disparities extraction, and
2. Disparity compensation (vergence control process).
Refer to Fig 4.1

There are a lot of algorithms that deal with disparities [16][17][18]. They are usually used to obtain a depth map. In disparity estimation for vergence control, what we need is an "overall" disparity estimation --- the disparity between the images. The whole image could be regarded as a single "big point". Our approach is Fourier phase-based approach. It is motivated by the Fourier translation property that a translation in spatial domain will result a translation in frequency domain that is direct proportional to spatial translation. When disparity exists in two images that are taken at the same time but in

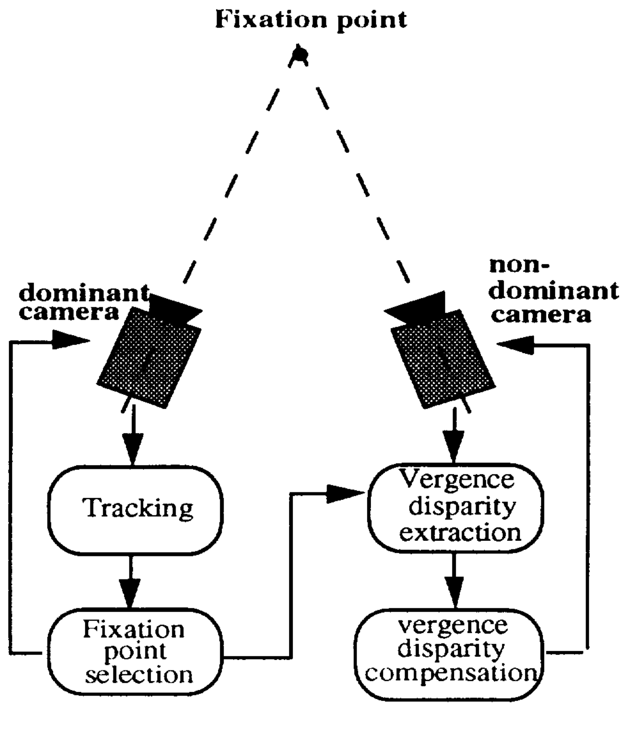


Fig 4.1 Different tasks in left and right camera for fixation

different cameras, we can regard the two images as taken consecutively in one camera and the disparity is due to the translation of the object. Thus, by calculating the phase difference of two "consecutive" image, we are able to determine the translation of the object in two consecutive images and then the actual disparities can be determined. Our approach is similar to [13] in that the two methods both use phase difference as a measure of disparity. But in [13], local disparities are important and this is why a local filter (Gabor filter) is involved since its goal is to obtain a depth map. In our approach, since we are only interested in "overall" disparity, the complicated gray-level images are used as binary images and treated as a single "large" point. Any local analysis is not necessary. Therefore, our approach is more suitable to vergence control.

The advantages of our approach over the existing approaches [3][5] for vergence control are:
1. We simplify the image processing --- gray-level images are used as binary images. The ideal and the seemingly unrealistic assumption (shifted version) becomes true in our approach.
2. The disparity is obtained directly as a function of the image property (Here only the contour is important.). It avoids the disadvantages contained in peak-finding method [12].
3. This approach is a robust estimation of disparity. Local occlusions and local intensity changes will not affect the "overall" disparity estimation.
4. It is simpler in that only phases are calculated. The computationally more expensive process of spectrum calculation is avoided while in [3][5] peaks are found in the spectrum analysis. Thus, presented approach is more suitable to real time application.

2. Vergence Disparity Measurement Based on Fourier Phase Difference

It is known that the Fourier phase difference between two consecutive images provides all the information required to obtain the relative displacement vector[15]. The most important advantage of using complex phase of Fourier transform in objection position detection is that a translation in the spatial domain directly corresponds to a phase shift in the spatial frequency domain. When an object is completely inside the image window, the relationship between position and fundamental frequency complex phase is linear [17][15]. More explicitly, the position and the fundamental frequency complex phase satisfy the following equation:

$$\Delta\text{position} = \frac{\text{window\_size}}{2\pi} * \Delta\text{phase} \qquad (4.1)$$

This equation can be directly obtained from the translation property of the Fourier transform represented by [24]:

$$f(x-x_0, y-y_0) \Leftrightarrow F(u, v)\exp[-j2\pi(ux_0 + vy_0)/N] \qquad (4.2)$$

where we only consider fundamental frequency ($u = v = 1$) and N is the window size.

If we regard the right image R(x, y) as an image that is taken in the left camera right after the image L(x, y) is taken and contribute the disparity to the shifts of the movement of the object with respect to the left camera, then, by calculating the fundamental frequency phase change in these two "consecutive" images, we are able to determine the disparity x_d and y_d. Once the disparities are determined, mapping them into vergence control system's reference input is not difficult.

It should be pointed out that the method introduced needs 2-D Fourier transform computation. One way to achieve faster processing is to use Fourier phase in conjunction with projection concept [15]. The use of projection is important because, in this way, it is possible to achieve 1-D processing and disparity x_d and y_d can be directly and separately obtained.

The projection of F(x, y) along *y*-direction onto *x-axis* perpendicular to *y-axis* is defined by [15]

$$F_y(x) = \int F(x, y)\,dy \qquad (4.3)$$

Similarly, we have projection of F(x, y) along *x*-direction onto *y*-axis:

$$F_x(y) = \int F(x, y)\,dx \qquad (4.4)$$

If we consider digital images, the integration should be represented as summation. Thus, equations (3.3) and (3.4) becomes:

$$F_j(i) = \sum_{j=0}^{h} F(i, j) \qquad (4.5)$$

$$F_i(j) = \sum_{i=0}^{w} F(i, j) \qquad (4.6)$$

where $h \times w$ is the window size and F(i,j) is quantized from F(x, y).

The algorithm below describes the procedure for vergence disparity extraction.

1. Determine an appropriate sized window such that the object is entirely within the window.
2. Get the projections of both images along *x*-direction and *y*-direction using:

$$L(i) = \sum_{j=0}^{h} L(i, j), \quad L(j) = \sum_{i=0}^{w} L(i, j) \qquad (4.7)$$

$$R(i) = \sum_{j=0}^{h} R(i, j), \quad R(j) = \sum_{i=0}^{w} R(i, j) \qquad (4.8)$$

3. Calculate their vertical and horizontal phases, which will be denoted by θ_L^i, θ_L^j, θ_R^i and θ_R^j, respectively.
4. The difference between the two pairs of phases will be

$$\Delta\theta^i = \theta_R^i - \theta_L^i \qquad (4.9)$$

$$\Delta\theta^j = \theta_R^j - \theta_L^j \qquad (4.10)$$

indicate the vertical and horizontal disparities according to (4.1).

$$x_d = \frac{h}{2\pi} * \Delta\theta^i \qquad (4.11)$$

$$y_d = \frac{w}{2\pi} * \Delta\theta^j \quad (4.12)$$

As we have known the coordinates of the point of fixation in the left image are x_{FPL}, y_{FPL} and the disparity is (x_d, y_d), the coordinates of the point of fixation in the right camera will be (x_{FPR}, y_{FPR}), which satisfy $x_{FPR} = x_{FPL} + x_d$ and $y_{FPR} = y_{FPL} + y_d$ and which will be the reference input to vergence servo system after kinematic transform.

V. Control Issues

The x_{REF} and y_{REF} are in terms of pixels. They should be transformed to other two values in terms of pan degrees or vergence degrees or tilt degrees, etc., through kinematic calculation since this is the only form the local controller can accept. As mentioned before, each degree of freedom has its own local controller., which are coordinated by the robot head platform control block. The presently implemented control algorithm is PD algorithm, i.e., the output of the controller is proportional to the error between reference input and system real output and the derivative of the error. This is a typical implementation for DC motor drive system and can be mathematically represented as:

$$u(t) = k_p * e(t) + k_d * \dot{e}(t) \quad (5.1)$$

where e(t) is the error between reference input $r_i(t)$ and system's real output y(t), i.e.,

$$e(t) = r_i(t) - y(t) \quad (5.2)$$

Different choices of the two parameters of the PD controller, k_d and k_p, will result different output response. the larger the k_p, the smaller the steady error but the larger the overshoot. The larger the k_d, the more sensitive the system, either speeding the response or resulting oscillation. So the two parameters are empirically selected such that the step response of the system is slightly under-damped to achieve fast response with small overshoot. The simulation of one of the controller's output is depicted in Fig 5.1.

VI. Conclusions

The design of an active vision system is given with emphasis on the ability to obtain accurate 3-D information and on the convenience for gaze control. Based on this design we discussed three problems involved in binocular system's gaze stabilization process.

In fixation point selection, we argued what kind of features can be chosen as fixation point candidates. In this paper, we select corner/edge-point as salient feature for fixation purposes. Studies in human visual behavior provide us with theoretical foundation based on which evaluation functions are formed to determine fixation point hierarchically from between the candidates. We should point out that appropriate target for fixation are chosen according to visual tasks the system is performing. Gaze control at the higher level can be

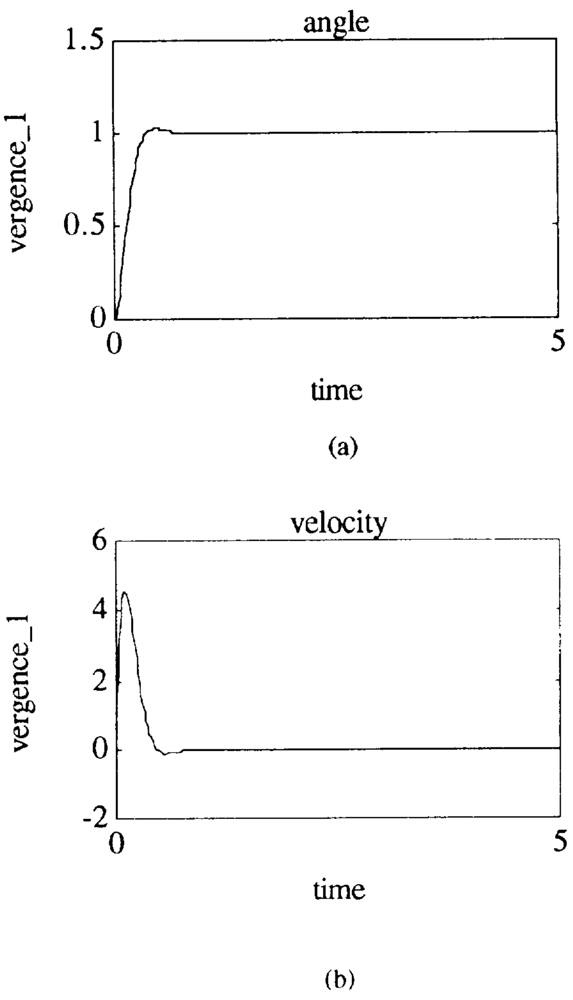


Fig 5.1 (a) Vergence servo output with small overshoot under step input. (b) The velocity of the output.

viewed as a resource management problem [3]. This is beyond the scope of this paper and is not taken into account. Here, we assume that corner/edge-point could be our appropriate target for fixation.

We characterized different tasks in left and right cameras for vergence control and used phase-based method to measure vergence error based on binarized images. This approach can robustly and efficiently extracts vergence disparities.

And in the last section we discussed some properties of the local controller based on PD algorithm.

Acknowledgment

This research has been partially supported in part by the grants DDM-9319208 and DDM-9210018 from the

National Science Foundation to Dr. Marefat. The provided support is gratefully appreciated.

VII. References

1. Ruzena Bajcsy, "Active Perception", Proceedings of the IEEE, Vol. 76, No. 8, August, 1988.1

2. University of Chicago, "Promising Directions in Active Vision", University of Chicago Techinical Report CS 91-27, November, 1991.

3. Thomas J. Olson and David J. Coombs, "Real-Time Vergence Control for Binocular Robots", Int. J. Computer Vision, 7:1, 1991.

4. A. Lynn Abbott, "A Survey of Selective Fixation Control for Machine Vision", IEEE Control Systems, August 1992.

5. David J. Coombs and Christopher M. Brown, "Cooperative Gaze Holding in Binocular Vision", IEEE Control Systems, June 1991 .

6. Rachid Deriche and Gerard Giraudon, "A Computational Approach for Corner And Vertex Detection", Int. J. Computer Vision, 10:2, 1993.

7. P.R. Beaudet, "Rotational Invariant Image Operator", *Int. J. Conf.* Pattern Recognition, pp. 579-583, 1978.

8. O.A. Zuniga and R. Haralick, "Corner Detection Using the Facet Model", IEEE CVPR Conference, pp. 30-37, 1983.

9. L. Kitchen and K. Rosenfeld, "Gray-level Corner Detection", Pattern Recognition Letter, Vol. 1, pp. 95-102, 1982.

10. H.H. Nagel, "Displacement vectors from Second-order Variations", Computer Vision, Graphics, and Image Processing, Vol. 21, pp. 85-117, 1983.

11. Mubarak A. Shah and Ramesh Jain, "Detecting Time-Varying Corners", Computer Vision, Graphics, and Image Processing Vol. 28, pp. 345-355, 1984.

12. Michael Jekin and John K. Tsotsos, "Techniques for Disparity Measurement", CVGIP: Image Understanding, Vol. 53, No. 1,, January 1991.

13. T.D. Sanger, "Stereo Disparity Conputation Using Gabor Filter", Biological Cybernetics, Vol 59, pp. 405 - 418, 1988.

14. D. De Vleeschauwer, "An Intensity-Based Coarse-to-Fine Approach to Reliably Measure Binocular Disparity", CVGIP: Image Understanding, Vol. 57, No. 2, March, 1993.

15. Mansur Kabuka, *et al,* "Robot Vision Tracking System", IEEE Transactions on Industrial Electronics, Vol. 35, No. 1, February, 1988.

16. Katsuhiko Ogata, Modern Control Engineering, Englewood Cliffs, 1970.

17. Gonzalez and Wintz, Digital Image Processing, Second Edition, Addison Wesley, 1987.

18 Huang Zhen Hua and Yu Qian, "A Direct Corner Detecting Algorithm", IEEE CVPR, 1986.

S22-63
207245

N94-30548 AIAA-94-1197-CP

VISION BASED OBJECT POSE ESTIMATION FOR MOBILE ROBOTS

Annie Wu, Clint Bidlack, Arun Katkere[*], Roy Feague, and Terry Weymouth
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109-2110
aswu@engin.umich.edu

Abstract

Mobile robot navigation using visual sensors requires that a robot be able to detect landmarks and obtain pose information from a camera image. This paper presents a vision system for finding man made markers of known size and calculating the pose of these markers. The algorithm detects and identifies the markers using a weighted pattern matching template. Geometric constraints are then used to calculate the position of the markers relative to the robot. The selection of geometric constraints comes from the *typical* pose of most man made signs; such as the sign standing vertical and the dimensions of known size. This system has been tested successfully on a wide range of real images. Marker detection is reliable, even in cluttered environments, and under certain marker orientations, estimation of the orientation has proven accurate to within 2 degrees, and distance estimation to within 0.3 meters.

Task description

Humans are very dependent on their sense of sight for navigation. People use both natural and man-made landmarks to help them determine where they are and which way they want to go next. What humans can do with the greatest of ease, however, can be very difficult for robots. Mobile robot navigation using visual sensors typically requires that the robot be able to obtain pose information from a camera image. This task often includes recognizing markers or other known objects in the image and calculating the object pose from the size and appearance.

There are several tasks that a robot navigating by vision must deal with: the robot must to be able to extract markers from a complex environment; the robot has to recognize these markers from many different points of view; and the robot must determine, from it's view of the marker, the pose (3D position and orientation) of the marker. In addition, for all practical purposes, the robot should be able to perform all of the above tasks relatively fast (less than a few seconds in most cases).

This paper describes a vision system that was implemented for the AAAI 1993 Robot Competition in Washington D. C. on July 11-16, 1993. All vision processing was performed onboard the robot using a 80486 PC DOS based computer. A complete description of the design of the University of Michigan entry can be found in [1].

The vision system is divided into a marker extraction and identification step, and a pose estimation step. Marker extraction finds predefined markers (black 'x's and '+'s on a white background) in the environment and determines their pose relative to the robot. Thus, a robot using this system should be able to navigate autonomously using visual sensors in a semi-constrained environment. The required geometric constraints are: the marker must stand vertical; the marker and camera contain no roll; the focal length of the camera and the camera's location relative to the robot are known; the robot is oriented in the plane perpendicular to the marker; and the width and height of the marker are known. Though these constraints may seem restrictive, they are typical of most man made signs such as traffic signs and office door markers.

Marker detection

The marker detection phase is composed of two main routines: the connected components routine and the marker identification routine. The detection phase must be both fast and accurate for the system to be useful for most real world tasks.

To maximize speed, we make only one pass through the entire image. During the pass, the image is thresholded and connected components are found and labeled. One pixel components are ignored and not labeled. Size thresholding then filters out most of the non-marker components. Only one pass is made through all possible connected components. Figure 1 shows sample output from this stage. The possible markers are outlined with a bounding box.

To identify or reject the remaining markers, a weighted pattern matching template is used. An nxn template matrix is created for each marker (see Figure 2). Increasing n increases the resolution of the template, but also increases the process time. We found $n = 7$ to be a good compromise. This weighted template indicates which areas are expected to be black and which ones white. The weights for our matrix are currently determined by trial and error, but we could easily replace these with machine gener-

[*]Currently at University of California, San Diego

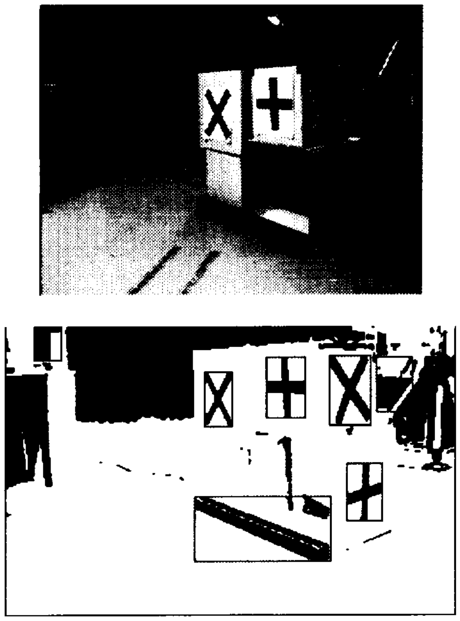


Figure 1: The first image is a typical input image. The second image shows the markers that are detected by the connected components routine. These markers will be identified as x, +, or neither.

1	1	-2	-8	-2	1	1
1	2	0	-1	0	2	1
-2	0	3	1	3	0	-2
-8	-1	1	8	1	-1	-8
-2	0	3	1	3	0	-2
1	2	0	-1	0	2	1
1	1	-2	-8	-2	1	1

x template

-4	-6	0	8	0	-6	-4
-2	-3	0	8	0	-3	-2
1	0	0	8	0	0	1
2	3	5	8	5	3	2
1	0	0	8	0	0	1
-2	-3	0	8	0	-3	-2
-4	-6	0	8	0	-6	-4

+ template

Figure 2: Weighted pattern templates for the x and the + markers. Positive values indicate expected black areas; negative areas are expected to be white. Certainty increases with magnitude.

b	w	w	w	w	b	b
b	b	w	w	b	b	w
w	b	b	b	b	w	w
w	w	b	b	b	w	w
w	w	b	b	b	b	w
w	b	b	w	w	b	b
b	b	w	w	w	w	b

Sample marker

$$Certainty_x = \frac{\sum_r \sum_c f_x(r,c)}{\sum_r \sum_c |x_{rc}|} = \frac{92}{96} = 0.9583$$

$$Certainty_p = \frac{\sum_r \sum_c f_p(r,c)}{\sum_r \sum_c |p_{rc}|} = \frac{50}{140} = 0.3571$$

$$Certainty = max(Certainty_x, Certainty_y \qquad (1)$$

$$f_x(r,c) = \begin{cases} |x_{rc}| & \text{if correct color} \\ 0 & \text{otherwise} \end{cases}$$

$$f_p(r,c) = \begin{cases} |x_{rc}| & \text{if correct color} \\ 0 & \text{otherwise} \end{cases}$$

Figure 3: Sample marker with calculated x and + certainty values. "b" indicates a black pixel; "w" indicates a white pixel. *x* refers to the x template; *p* refers to the + template. *r* counts rows; *c* counts columns. For this example, the program is 95.8% certain that the sample marker is an x and 35.7% certain that it is a +.

ated weights if a learning program were implemented. The marker template which a component most resembles is selected as the "guess" for that component. The program generates a certainty measure with each guess (see Figure 3) and uses this measure to accept or reject the guess.

Each marker can have one or more templates. The additional templates may be used to improve marker recognition from other views.

Two types of heuristic information is also used in identifying the markers. Some heuristics were known before the program was written. Knowing that all +'s have a vertical line down the center of the bounding box, no matter what the robot's relative position, has strongly emphasized the importance of the center line in the template. Other heuristics were not learned or incorporated until after the program had been tested. Diagonal lines often scored high enough certainty values to be considered x's. Adding a specific test to verify that each possible x is not a diagonal line solved this problem.

Pose estimation

The three dimensional position and orientation (pose) of the markers is also determined. Such information is useful for performing further analysis.

One possible application of the pose estimation algorithm is the detection of road signs. Once a sign's pose is calculated the pixels corresponding to the sign can be mapped to an orthographic projection. Since virtually all character recognition algorithms assume an orthographic projection, this would allow for *much* improved character recognition.

For the robot competition, the pose of the markers also represents the pose of the box to which the marker is attached. One phase of the competition requires the robot to autonomously move the box from one location to another. The marker pose is used to guide the robot to the box such that the box can be pushed to the appropriate location.

Geometric constraints are used to calculate the position of the markers relative to the robot. First, the marker is expected to be mounted on a planar surface and that the four corners of the marker are detected from the low level image processing (marker extraction and identification). The markers dimensions are also know in advance. Second, the marker is standing vertical. As mentioned before, this is not an unreasonable constraint as many man made signs stand vertical. Finally, the calibration parameters of the camera are known, including orientation of the camera relative to the robot and the camera's focal length. Also, there should be minimal* camera roll (rotation about the Z axis).

These geometric constraints form a set of 24 equations in 18 unknowns defining the position of the four corners of the markers. This provides an overconstrained set of equations which is solved using the method of least squares. The final result are the 3D position of the four corners of the markers. For the given application, the orientation of the markers and the distance to the center of the marker are calculated from the four 3D positions. These two values are used by the robot to navigate to the markers so that more accurate identification and pose calculations can be made.

Utilizing Geometric Constraints

Figure 4 depicts the geometry of the imaging process with the bounding box of a '+' marker being mapped to the image plane. Both the width (w) and height (h) of the markers are known. The three dimensional unit direction vectors $\vec{n1}$, $\vec{n2}$, $\vec{n3}$, and $\vec{n4}$, which are directed from the known focal center of the camera $\vec{F}$ towards the unknown marker position vectors $\vec{P1}$, $\vec{P2}$, $\vec{P3}$, and $\vec{P4}$, are calculated. This calculation is feasible given the position of the focal center of the camera $\vec{F}$, and given the four sensor plane 2D position vectors $\vec{p1}$, $\vec{p2}$, $\vec{p3}$, and $\vec{p4}$. These 2D vectors correspond to the mapping of the corners of the markers onto the sensor plane. Due to the imaging process, distances $d1$, $d2$, $d3$, and $d4$ are unknown (where dn is the distance in 3D space from $\vec{pn}$ to $\vec{Pn}$). Figure 5 shows the coordinate frame assigned to the camera's sensor plane Ψ_s and its relation to camera's 3D coordinate frame Ψ_c, the image coordinate frame Ψ_i, and the robot coordinate frame Ψ_r.

*Current experimentation indicate that both a marker tilt and marker (or camera) tilt of up to 10 degrees do not significantly effect the calculation of the position of the marker. In addition, the effects on the orientation also seem negligable relative to other errors. Further testing is being performed.

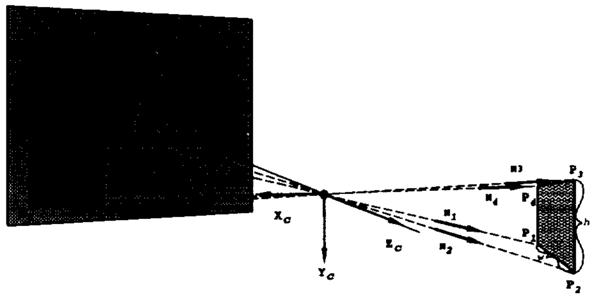


Figure 4: Mapping of objects onto the image plane.

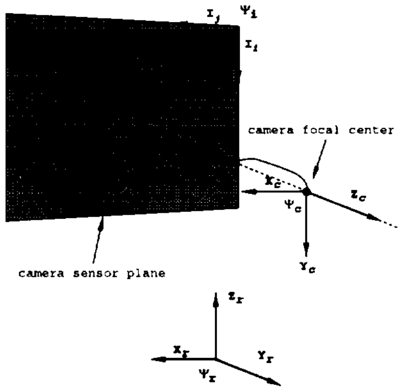


Figure 5: Locations of coordinate frames

It is assumed that the camera focal length is known and that the pose of the camera relative to the robot is also know. Then all points are transformed to the robot coordinate frame Ψ_r. This results in the following equations of known vectors:

$$\vec{n1} = [-p1_x, -p1_y, f] \quad (2)$$

$$\vec{n2} = [-p2_x, -p2_y, f] \quad (3)$$

$$\vec{n3} = [-p3_x, -p3_y, f] \quad (4)$$

$$\vec{n4} = [-p4_x, -p4_y, f]. \quad (5)$$

The vector equations with unknowns are:

$$\vec{P1} = d1 \times \vec{n1} \quad (6)$$

$$\vec{P2} = d2 \times \vec{n2} \quad (7)$$

$$\vec{P3} = d3 \times \vec{n3} \quad (8)$$

$$\vec{P4} = d4 \times \vec{n4}. \quad (9)$$

In addition the following constraint equations arise given the marker is standing vertical and that the camera and marker have no roll (rotation about the Z axis). Here $d1$, $d2$, $d3$, and $d4$ are the distances from the camera focal center to the unknown 3D points $P1$,

$P2$, $P3$, and $P4$.

$$P2_x = d1 \times n1_x + w \times nw_x \tag{10}$$
$$P2_y = d1 \times n1_y + w \times nw_y \tag{11}$$
$$P3_x = d4 \times n4_x + w \times nw_x \tag{12}$$
$$P3_y = d4 \times n4_y + w \times nw_y \tag{13}$$
$$P4_z = d1 \times n1_z + h \tag{14}$$
$$P3_z = d2 \times n2_z + h \tag{15}$$
$$d1 \times n1_z = d2 \times n2_z \tag{16}$$
$$d4 \times n4_z = d3 \times n3_z \tag{17}$$
$$P4_x = P1_x \tag{18}$$
$$P4_y = P1_y \tag{19}$$
$$P3_x = P2_x \tag{20}$$
$$P3_y = P2_y. \tag{21}$$

These equations can be expressed as an overconstrained system of linear equations with the above 24 equations and the 18 unknowns of $d1$, $d2$, $d3$, $d4$, $\vec{P1}$, $\vec{P2}$, $\vec{P3}$, $\vec{P4}$, and $\vec{nw}$. The two dimensional unit vector $\vec{nw}$ has an x and y component. nw_x corresponds to the x component of the vector pointing from $\vec{P1}$ to $\vec{P2}$, and nw_y corresponds to the y component of this vector. There is no z component to $\vec{nw}$ since the markers, and the camera, are assumed to have no roll.

Equations 2 thru 21 result in the matrix equation

$$\vec{y} = A\vec{x}, \tag{22}$$

where $\vec{y}$ is the 24 element known vector

$$\vec{y}=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,H,H,0,0,0,0) \tag{23}$$

and $\vec{x}$ is the 18 element unknown vector

$$\vec{x}=(P1_x, P2_x, P3_x, P4_x, P1_y, P2_y, P3_y, P4_y, P1_z, P2_z, P3_z, P4_z, d1, d2, d3, d4, nwx, nwy) \tag{24}$$

and the matrix $\mathbf{A}$ is the following:

$$\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&-n1x&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&-n2x&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&-n3x&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&-n4x&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&-n1y&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&-n4x&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&-n3y&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&-n4y&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&-n1z&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&-n2z&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&-n3z&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&-n4z&0&0\\
0&-1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&-1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&-1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-n3z&n4z&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&n1z&-n2z&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&-n2z&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&-n1z&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&-n1x&0&0&0&-W&0\\
0&0&0&0&0&1&0&0&0&0&0&0&-n1x&0&0&0&0&-W\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&-n4x&-W&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&-n4y&0&-W
\end{bmatrix}$$

Results

The accuracy of the pose estimation algorithm is measured by the error between the estimated and true marker distance and orientation. Robustness refers to the program's ability to detect markers and make reasonable pose estimations in complex situations such as cluttered images, tilted camera, uneven floor, etc. A set of experiments have been performed which test these measures.

The testing of this vision system has produced promising results. Marker extraction and identification is very accurate, even in cluttered images. Markers can be extracted at orientations of up to 60 degrees. Pose estimation is possible in the range of one to seven meters. Distance can be determined to within .2 meters when the marker is at an orientation of 50 degrees. Marker orientation can be as accurate as 1 degree; the ground truth measurements of orientation is approximately 1 degree, so any error at this resolution could be a factor of either the vision system or the ground truth measurements of the marker orientation. These results were obtained on low resolution images of 315 by 200 pixels. Figure 6 shows two sample images with the calculated marker pose projected onto the images.

The system should be able to extract only and all markers in an image. If a tradeoff must be made, then it is prefered to that non-markers be identified as markers. The robot can then approach *false* markers and perform further analysis to determine that indeed this marker is not a false positive. To make

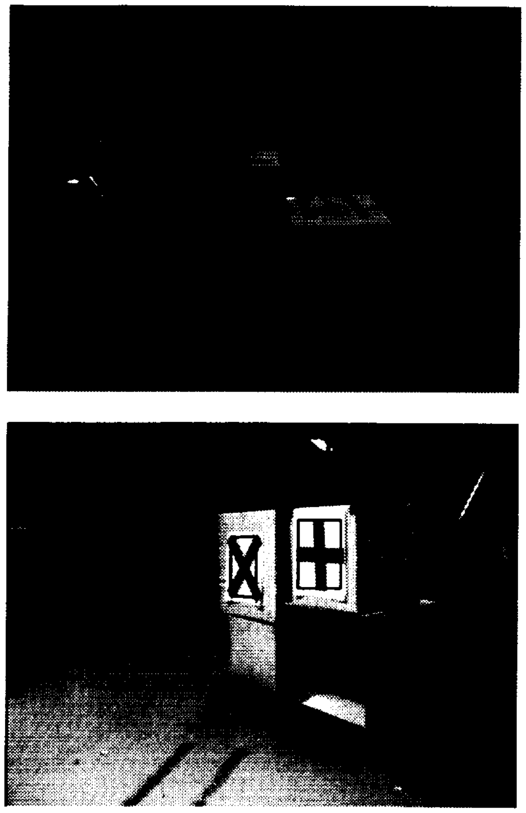


Figure 6: Two sample images with the calculated marker pose projected on top.

such an analysis more tractable, the vision system should output a confidence value with each marker sighting, which would be used by the robot to determine which markers need further analysis. With each classification, the marker detection algorithm generates a certainty value as given in equation 1, and the pose estimation algorithm generates δ, the residual from the least squares fit as

$$\delta = A\vec{x} - \vec{y} \tag{25}$$

These two values, *residual* and *certainty*, are available to the robot to help determine how to accept the marker and its pose.

The experiments involved processing of 42 images, each having two to four markers. Only once did the marker detection step identify a non-marker object as a marker (a false positive). The program only missed existing markers when oriented at angles greater than 50 degrees and often detects markers up to 70 degrees.

The original purpose of the marker size threshold was to eliminate obvious non-marker components as soon as possible and reduce the number of connected components that are processed by the marker identification routine. If the user can set the threshold to limit the size of the markers to a small range, fewer extraneous components are then processed by the marker identification routine, reducing the chance of false positives. Unfortunately, a small range also limits the distance at which markers can be recognized. During testing, it was found that a narrow size threshold was not crucial for accurate identification. Marker sizes in the distance images ranged from about 50 pixels at seven meters to over 1000 pixels at one meter. Even with such a wide size range, the program returned a false positive only once, while successfully finding over 100 markers in 42 test images.

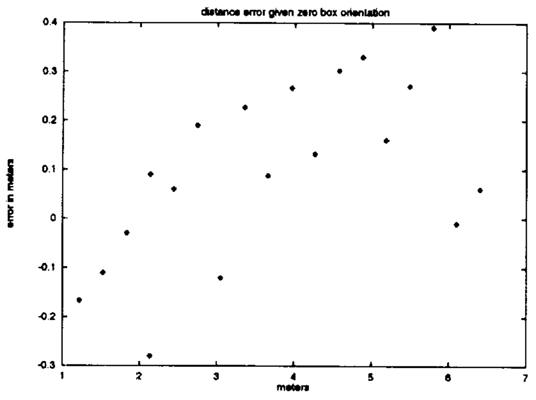


Figure 7: Plot of the error in calculated distance as actual distance increases and with zero box orientation.

Figures 7, 8, and 9 are plots of some of the experiments. The first plot displays the calculated distance error as a function of distance to the marker. All tests resulted in an error of less than 0.4 meters and over half being less than 0.2 meters. As expected, the results show that the error generally increases as the distance from the object increases. The main exception being the two data points around 6 meters that have a very small error. More data points are needed to determine if this is the not due to some unforeseen anomaly of the algorithm, or just chance, as we suspect it is.

Figure 8 displays the results from the experiment to test the distance accuracy as a function of marker orientation. The marker detection algorithm can not reliably segment markers at orientations above 60 degrees, hence the orientation plots only extend from zero to 60 degrees. An orientation of 0 degrees corresponds to the marker being perpendicular to the imaging plane. All these tests were from a distance of 2.16 meters. The distance error is within 0.13 meters with a marker orientation between zero degrees and 50 degrees.

Figure 9 represents the experiment to test the orientation calculation accuracy as a function of marker orientation. All the tests were from a distance of 2.16 meters again. This plot displays the interesting feature that the error is minimal between 30 and 60 degrees. Also, the error increases from 30 degrees back to 0 degrees of marker orientation. This effect is due to the perspective transformation; when objects are perpendicular to the imaging plane, small perturbations in the objects orientation make even smaller changes in the view as mapped to the imaging plane. The small perturbation effects increase as the angle increases (object becoming less perpendicular to the imaging plane). This effect causes fairly large

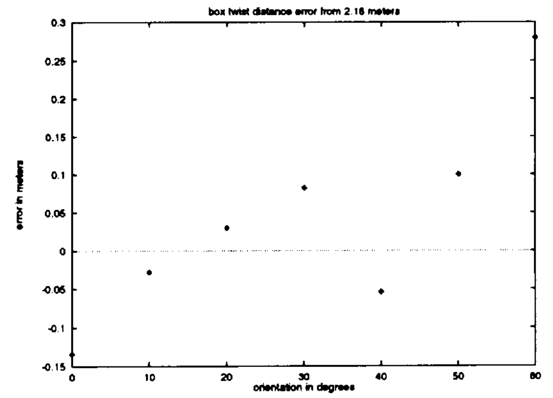


Figure 8: Plot of the error in calculated distance as box orientation changes and at constant distance of 2.16 meters.

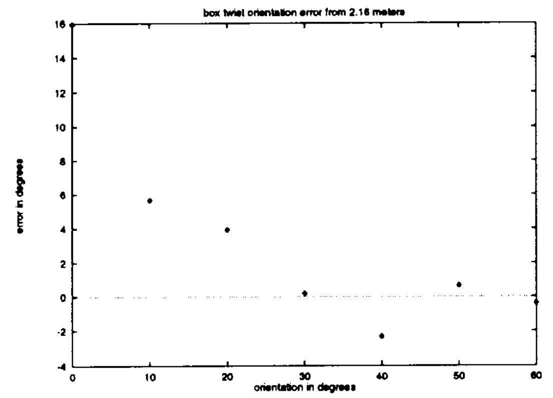


Figure 9: Plot of the error in calculated box orientation as actual box orientation increases.

changes in the orientation of the markers (when the object is almost perpendicular to the imaging plane) to account for small changes in the mapping of the marker onto the image plane. Hence, small changes in marker orientation go unnoticed by the algorithm when the object orientation is much less than 30 degrees. Perhaps more appropriately, small errors in the pixel locations of the four corners of the marker result in large changes in the computed object orientation when orientations are less than 30 degrees. Errors in the marker detection algorithm become more crucial under small orientation angles, with our experimental results showing this to be true as well. This *marker orientation sensitivity* can be shown analytically as well. Figure 10 shows a two dimensional representation of the problem. For our experiments, the variables f, D, and L are known and have values of 0.0085 meters, 2.16 meters and 0.23 meters respectively. f corresponds to the camera focal lenght, D the distance from the camera to the marker, and L the width of the marker. The following equations are basic geometry equations from Figure 10:

$$L_p = Dl/f \qquad (26)$$

$$\theta = 180 - atan(f/l) \qquad (27)$$

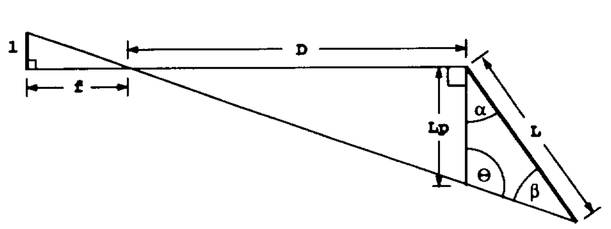


Figure 10: The two dimensional representation of the orientation of the marker relative to the imaging plane.

$$\beta = arcsin((Dl/(fL))sin(\theta_1)) \qquad (28)$$

$$\alpha = 180 - \theta - \beta. \qquad (29)$$

Now solving for α as a function of l,

$$\alpha(l) = \arctan(\frac{f}{l}) - \arcsin(DL^{-1}\frac{1}{\sqrt{1+\frac{f^2}{l^2}}}), \qquad (30)$$

and its derivative with respect to l is

$$\frac{d\alpha(l)}{dl} = -fl^{-2}\left(1+\frac{f^2}{l^2}\right)^{-1} - Df^2 * \qquad (31)$$

$$\frac{1}{\sqrt{1-D^2L^{-2}\left(1+\frac{f^2}{l^2}\right)^{-1}}}L^{-1}\left(1+\frac{f^2}{l^2}\right)^{-3/2}l^{-3}.$$

Figure 11 represents the plot of $\alpha(l)$ for values of l from zero to $\frac{L_p f}{D}$, and Figure 12 is a plot of $\frac{d\alpha(l)}{dl}$ for the same range of l. Notice the sharp *knee* in $\frac{d\alpha(l)}{dl}$ at $l \approx 0.0007$. This shows that for $l < 0.0007$ meters the magnitude of the rate of change of α with respect to l is fairly constant and small. However, for $l > 0.0007$ meters, the magnitude of this rate of change increases very rapidly, meaning that small perturbations in the length l (the measured width of the marker) result in large changes in the marker orientation. When the marker detection process introduces small spatial measurement errors, for example, due to quantization of the image and the due to the marker segmentation process itself, then the resulting estimated orientation errors may be very large when $l > 0.0007$ meters. This corresponds to the experimental results as shown in Figure 9. Also, from the plots in Figure 12 and 11, the location of the *knee* at 0.0007 meters corresponds to an angle of approximately 0.6 radians or 34 degrees. This in turn, corresponds to the experimental findings that the orientation error increases for values of marker orientation less than approximately 30 degrees.

Conclusions

Results from this project indicate that it is possible to obtain useful pose information from a camera image in real time on a general purpose computer such as a 80486 based PC. Additional tests on the sensitivity of pose estimation to various parameters such as focal length value perturbations and marker size are planned. In addition, we will be studying the tradeoffs between process time (i.e. image resolution) and accuracy.

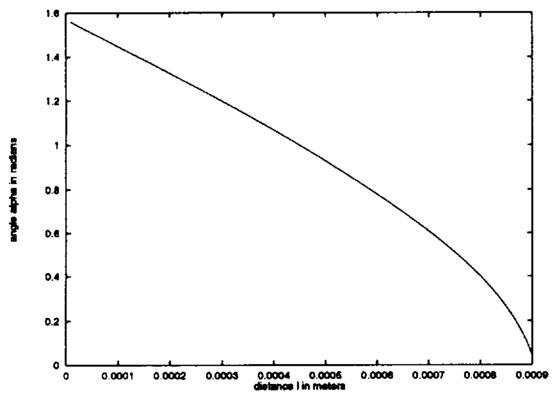


Figure 11: Plot of $\alpha(l)$.

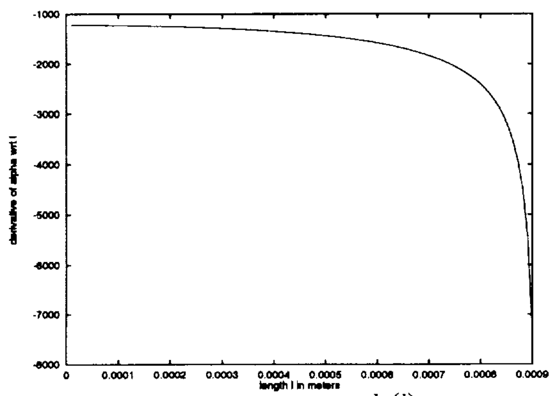


Figure 12: Plot of $\frac{d\alpha(l)}{dl}$.

References

[1] David Kortenkamp, Marcus Huber, Frank Koss, William Belding, Jaeho Lee, Annie Wu, Clint Bidlack, and Seth Rogers. Mobile robot exploration and navigation of indoor spaces using sonar and vision. In *Conference on Intelligent Robots in Field, Factory, Service and Space*, March 1994.

6044 N94- 30549

p. 6

AIAA-94-1199-CP

UNSUPERVISED TEXTURE IMAGE SEGMENTTATION BY IMPROVED NEURAL NETWORK ART2

Zhiling Wang, G. Sylos Labini, R. Mugnuolo and Marco De Sario[†]

Center for Space Geodesy, Italian Space Agency, P.O. Box 11
75100 Matera, Italy
Fax:+39-835-339005 Tel:+39-835-3779 Email:zhiling@asimt0.mt.asi.it

[†]*Dept. of Electronic Engineering, University of Bari, Via Re David,200*
70125 Bari , Italy

Abstract

We here propose a segmentation algorithm of texture image for computer vision system on space robot. An improved *Adaptive Resonance Theory* (ART2) for analog input patterns is adapted to classify the image based on a set of texture image features extracted by a fast *Spatial Gray Level Dependence Method* (SGLDM). The nonlinear thresholding functions in input layer of the neural network have been constructed by two parts: firstly to reduce the effection of image noises on the features, a set of sigmoid functions is chosen depending on the types of the feature; secondly, to enhence the contrast of the features, we adopt *fuzzy mapping functions* The cluster number in output layer can be increased by an autogrowing mechanism constantly when a new pattern happens. Experimental results and orginal or segmented pictures are shown, including the comparison between this approach and K-means algorithm. The system written by C language is performed on a SUN-4/330 sparc-station with an image board IT-150 and a CCD camera.

1. Introduction

Segmentation and classification of textured images have been considerable attention to contain significant discriminatory information for image segmentation in a variety of application, such as terrain classification, military surveillance and recognition, remote sensing images and biomedical image analysis[1]. Although texture is a fundamental characteristic of images , the complexity involved in its quantification has presented its effective incorporation into the segmentation process.

in this paper, the neural network of an improved Adaptive Resonance Theory (ART2) is presented to segment an image consisting of several regions with different textures. Artificial neural networks offer several advantages over conventional classification techniques, due to their high computation rate, great degree of fault tolerance and unsupervised ability. The number of researches have engaged on the researchment by neural networks[20~31].

In this paper, section 2 defines the texture feature types which are derived from co-occurrence matrixes and selection of maximum and minimum measure window for feature extraction of the texture image. Section 3 describes an approach of improved ART2 neural network with alterable competitive layer (F_2 layer). The nonlinear thresholding fuction in F_1 layer is displaced by a fuzzy mapping function. Section 4 shows the results of experiments and illustration.

2. Feature extraction of texture image

Whether the segmentation of texture image is good or not depends on the extraction of texture features. There are number of the approaches to have been developed for feature extraction of the texture image: Fourier power spectrum method (FPSM)[3],spatial texture energy[5], Markov random field model[7], Gibbs random field model[8,9], zero-sum filter masks [14], gray level run length method (GLRLM) [10], spatial gray level dependece method (SGLDM)[2], gray level difference method (GLDM) [11], and other methods [6,12,13,15]. Some of these methods belong to statistical method, others to structural one. Among them, spatial gray level dependence method, which is introduced by Haralick *et al.* in their paper[2], is one of the most successful statistical representation for the texture. The feature measurement from co-occurrence matrices in the SGLDM is rather similar to the knowledge captured by the human eyes, and provides a convenient way to represent the properties of object textures. Weszka *et al.* experimentally compared feature on terrain images and found that SGLDM is more powerful than the GLDM, GRLM, and FPSM [1]; Ohanian *et al.* also pointted that the features by SGLDM were better than Markov random field, multi-channel filtering features, and fractal based features[16]. It is known, however, that the SGLDM requires much processing time and great number of memory. Only for mean probability distribution, 2^{34} times of multiplication in the SGLDM are done when a measured image is a size 64×64 with gray level 128, and the tendency will be raised at exponent rate with the enlargement of the image size, particularly, the increase of gray-level number.

In this paper, we use a set of simplifed equations based on a fact that rows or columns around the current pixel are included or excluded almostly at the same time while the measured window is displaced in the horizontal or vertical direction of the image, so we could make the equation be simplifed viewing from the pixels of rows(columns) both excluded and included from a window rather than a pixel method [4]. Some calculations are done one time in a row or column instead of one in a pixel, so algorithm in the paper consumes much less time than Harilick's method.

We defined a co-occurrence matrix of relative frequencies with which two pixels separated by distance d at a specified angle occur on the image, one with gray level i and the other with gray level j.

A distance of one pixel, i.e. the measuring window slides over the image in one step length[1] in both horizontal and vertical direction and angle quantixed to 45° intervals, or 0°, 45°, 90°, and 135° will be used. We give a set of simplified equations:

(1) Mean

$$m_k = m_{k-1} + \frac{1}{N}\sum(i^+_{k,r} + j^+_{k,r} - i^-_{k,r} - j^-_{k,r}) \tag{1}$$

(2) Variance

$$\sigma_k^2 = \sigma_{k-1}^2 + \frac{1}{N}\sum[(i^+_{k,r})^2+(j^+_{k,r})^2-(i^-_{k,r})^2-(j^-_{k,r})^2]+m^2_{k-1}-m^2_k \tag{2}$$

(3) Correlation

$$C_k = [C_{k-1}\sigma^2_{k-1} + \frac{2}{N}\sum(i^+_{k,r}j^+_{k,r} - i^-_{k,r}j^-_{k,r}) + m^2_{k-1} - m^2_k]/\sigma^2_k \tag{3}$$

(4) Energy

Let

$$L^- = \frac{N_{k-1}}{N_{k-1}-2}, \qquad J^- = 1/(N_{k-1}-2)$$

$$L^+ = \frac{N_{k-1}}{N_{k-1}+2}, \qquad J^+ = 1/(N_{k-1}+2)$$

$$E^-_k = (L^-)^2E^-_{k-1} + (J^-)^2(a - 4M^-_{k-1}) \tag{4}$$

$$E^+_k = (L^+)^2E^+_{k-1} + (J^+)^2(a + 4M^-_{k-1}) \tag{5}$$

where

$$a = \begin{cases} 2 & \text{if } i \neq j \\ 4 & \text{if } i = j \end{cases} \tag{6}$$

(5) Entropy

$$EP^-_k = L^-EP^-_{k-1} - L^-\log(L^-) - A^- \tag{7}$$

$$EP^+_k = L^+EP^+_{k-1} - L^+\log(L^+) - A^+ \tag{8}$$

$$A^- = \begin{cases} 2J^-(M_{k-1}(i^-_{k,r}, j^-_{k,r}) - 1)\log(L^-(M_{k-1}(i^-_{k,r}, j^-_{k,r}) - 1)+ & \\ 2J^-M_{k-1}(i^-_{k,r}, j^-_{k,r})\log(L^-M_{k-1}(i^-_{k,r}, j^-_{k,r})) & \text{i} \neq j \\ J^-(M_{k-1}(i^-_{k,r}, j^-_{k,r}) - 2)\log(L^-(M_{k-1}(i^-_{k,r}, j^-_{k,r}) - 2)+ & \\ J^-M_{k-1}(i^-_{k,r}, j^-_{k,r})\log(L^-M_{k-1}(i^-_{k,r}, j^-_{k,r}) & \text{i=j} \end{cases}$$

$$A^+ = \begin{cases} 2J^+(M_{k-1}(i^+_{k,r}, j^+_{k,r}) + 1)\log(L^+(M_{k-1}(i^+_{k,r}, j^+_{k,r}) + 1) & \\ 2J^+M_{k-1}(i^+_{k,r}, j^+_{k,r})\log(L^+M_{k-1}(i^+_{k,r}, j^+_{k,r}) & \text{i} \neq j \\ J^+(M_{k-1}(i^+_{k,r}, j^+_{k,r}) + 2)\log(L^+(M_{k-1}(i^+_{k,r}, j^+_{k,r}) + 2) & \\ J^+M_{k-1}(i^+_{k,r}, j^+_{k,r})\log(L^+M_{k-1}(i^+_{k,r}, j^+_{k,r}) & \text{i=j} \end{cases}$$

[1]Ordinarily, rather than using a single displacement because small values for step length d yield the best results for the extraction of image features proved by Weszka *at. al.*[3]

(6) Contrast

$$T_k = T_{k-1} + \frac{2}{N}\sum[(i^+_{k,r} - j^+_{k,r})^2 - (i^-_{k,r} - j^-_{k,r})^2)] \tag{9}$$

(7) Homogeneity

$$H_k = H_{k-1} + \frac{2}{N}\sum\{(1+(i^+_{k,r}-j^+_{k,r})^2)^{-1}-(1+(i^-_{k,r}-j^-_{k,r})^2)^{-1}\} \tag{10}$$

where the $\sum$ is the $\sum_{r=1}^{L}$, the L stands for the length of the row or column, i.e. the wide of measuring square window, the M(i,j) is the element of a co-occurrence matrix, superscipts "+" and "-" express for a pixel (x,y) included or excluded from the window. The equations for both energy and entropy features are used to the case considering a pixel included or excluded from the window because of the nonlinear decomposition for square and logarithm functions

3. Improved ART2

Connectionist classification used here is called Adaptive Resonance Theory(ART) [24~27]. In general, ART is divided into two types depending on input patterns. ART1 is applied to solve binary input problem, ART2 is available to both binary and analog inputs. In the paper, the ART2 is used to classify the texture image because the 20 features (five for each angle) belong to gray-scale patterns.
The classifier in the ART2 consists mainly of two subsystems: the attentional subsystem and the orienting subsystem. The former is composed of the Short-Term Memory (STM) and Long-Term Memory (LTM) elements.

3.1 Short-Term Memory (STM)

The F_1, the input representation field, and F_2, the category representation field(competitive mechanism), are the two STM main components.

F_1 is composed of three layers with STM activation equations as (see Fig. 1)

$$p_i = u_i + \sum g(y_j)Z_{ji} \tag{11}$$

$$q_i = \frac{p_i}{e + \| p \|} \tag{12}$$

$$v_i = f(x_i) + bf(q_i) \tag{13}$$

$$u_i = \frac{v_i}{e + \| v \|} \tag{14}$$

$$w_i = I_i + au_i \tag{15}$$

$$x_i = \frac{w_i}{e + \| w \|} \tag{16}$$

where a,b, and e are constants, y_i is the STM activation of the Jth F_2 neuron, $\| \|$ is the L_2 norm, $f()$ is a nonlinear threshold function:

$$f(x) = \begin{cases} 0 & for\ \theta \leq x \leq \theta \\ 2(\frac{x-\theta}{\beta-\theta})^2 & for\ \theta \leq x \leq \alpha \\ 1 - 2(\frac{x-\beta}{\beta-\theta})^2 & for\ \alpha \leq x \leq \beta \\ 1 & for\ x \geq \beta \end{cases} \tag{17}$$

where the feature noises are suppressed by seting *f(x)* to zero when $0 \leq x < \theta$. The fuzzy mapping function is used to enhence the contrast among the features, and makes the input patterns classified

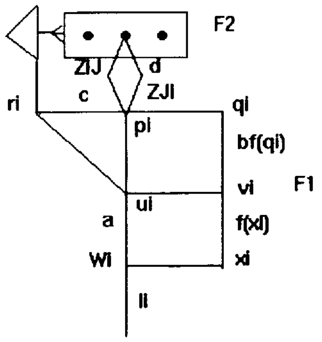


Fig. 1. Typical ART2 architecture [25,26,27]

more easily. The normalization mechanism keeps the pattern from saturation in spite of the constant presence of the pattern during the learning process. The F_1 layer provides internal feedback and a correlation between normalized bottom-up and top-down patterns to stabilize all activities in the STM before transmitting the output of the F_1 layer to the F_2 layer.

3.2 The search phase

In the F_2, a competitive mechanism is used to choose a winning neuron. Firstly, the input pattern of the F_1 is applied to the bottom-up adaptive filter by the bottom-up adaptive weight Z_{ij}.

$$T_j = \sum p_i Z_{ij} \quad for\ j = 1, 2, \ldots, K \tag{18}$$

where K is the total number of existing categories in the F_2, then the vector **T** is put in the order from minimum value to maximum one. We here suppose the Jth neuron in the F_2 is selected if this neuron becomes maximally active one among the neurons not to be reset in the trial, i.e.

$$T_J = max(T_j) \quad j = 1, 2, \ldots, K_J \tag{19}$$

where K_J is the total number of the categories not to be used, then only winning neuron in the F_2 has nonzero outputs.

$$g(T_j) = \begin{cases} d & \textit{if the Jth } F_2 \textit{ neuron is the winner based on} \\ & max(\sum p_i Z_{ij}) \textit{ and it has not been reset in the trial} \\ 0 & \textit{otherwise} \end{cases} \tag{20}$$

The top-down pattern $g(T_j)$ is then feedback to the F_1 by top-down adaptive weight Z_{ji} and compared to the original bottom-up pattern to see if a correct match has been made by an activated orienting subsystem.

3.3 Orienting subsystem

The orienting subsystem helps to directly search for the categories in the F_2. When the subsystem is activated, the bottom-up pattern vector **p** and the top-down pattern vector **u** are utilized to calculate the degree of match (vector **r**)

$$r_i = \frac{u_i + cp_i}{e + \| u \| + \| cp \|} \tag{21}$$

if the choise in the F_2 is correct, i.e.

$$(\| r \| > \rho \tag{22}$$

where ρ stands for the vigilance factor or match sensitivity parameter. At this time, adaptive resonance is considered to have occured and entered to the categories in the Long-Term Memory (LTM). If the choice is incorrect, another neuron with maximum output value among the existing neurons not to be selected will be selected as a possible winner candidate. The new candidate may cause yet another mismatch, hence another reset happens and the selection of yet another neuron, eventually, either the bottom-up pattern will be placed in an existing category or learned as the first example of a new category in the F_2 layer. It is possible for an autogrowing mechanism to be activated to create a new catogory if no category in the F_2 could be used to save the new one.

3.4 The Long-Term Memory (LTM)

The LTM is made up of two components, the bottom-up adaptive weight Z_{ij} and the top-down adaptive weight Z_{ji}. When the match operation in the orienting subsystem occures successfully, the bottom-up and top-down weights should be adjusted. The weights can been obtained easily by

$$Z_{iJ} = \frac{u_i}{1 - d} \tag{23}$$

$$Z_{Ji} = \frac{u_i}{1 - d} \tag{24}$$

The procedure in the improved ART2 can be summarized as:
Step 1. Initialize bottom-up and top-down adaptive weights Z_{ij} and Z_{ji} in the LTM.
Step 2. Apply a new input pattern.
Step 3. Stabilize the output vectors **u**(or **p**) of the F_1 layer by repeated operating Eqs. 11 ~ 16, including noise reduction and contrast enhencement by a nonlinear thresholding function and fuzzy mapping function.
Step 4. Compute the output vector p by Eq. 18.
Step 5. Select a winner neuron by Eqs. 18 and 19 if neurons not to be selected exist, else go to step 7.
Step 6. Apply Eq. 21 to determine whether the selected top-down winner pattern matches the bottom-up input **u** within a certain acceptance level of vigilance. if Eq. 21 is not true, the selected winner neuron in the F_2 is disabled and return to step 5 in order to choose another winner neuron ; else go to step 8;
Step 7. Autogrowing mechanism is activated to create a new category.
Step 8. Only adjust the bottom-up and top-down adaptive weights with respect to the matched winner neuron by Eqs. 23 and 24.
Step 9. Before taking the next new input pattern, neuron which has been disable int step 6 will be enabled. The process return to step 2 if a new input pattern at least exists, else exit the system.

Viewing from the improved ART2 algorithm, if the network for an input patten has learned previously to recognize the pattern, then

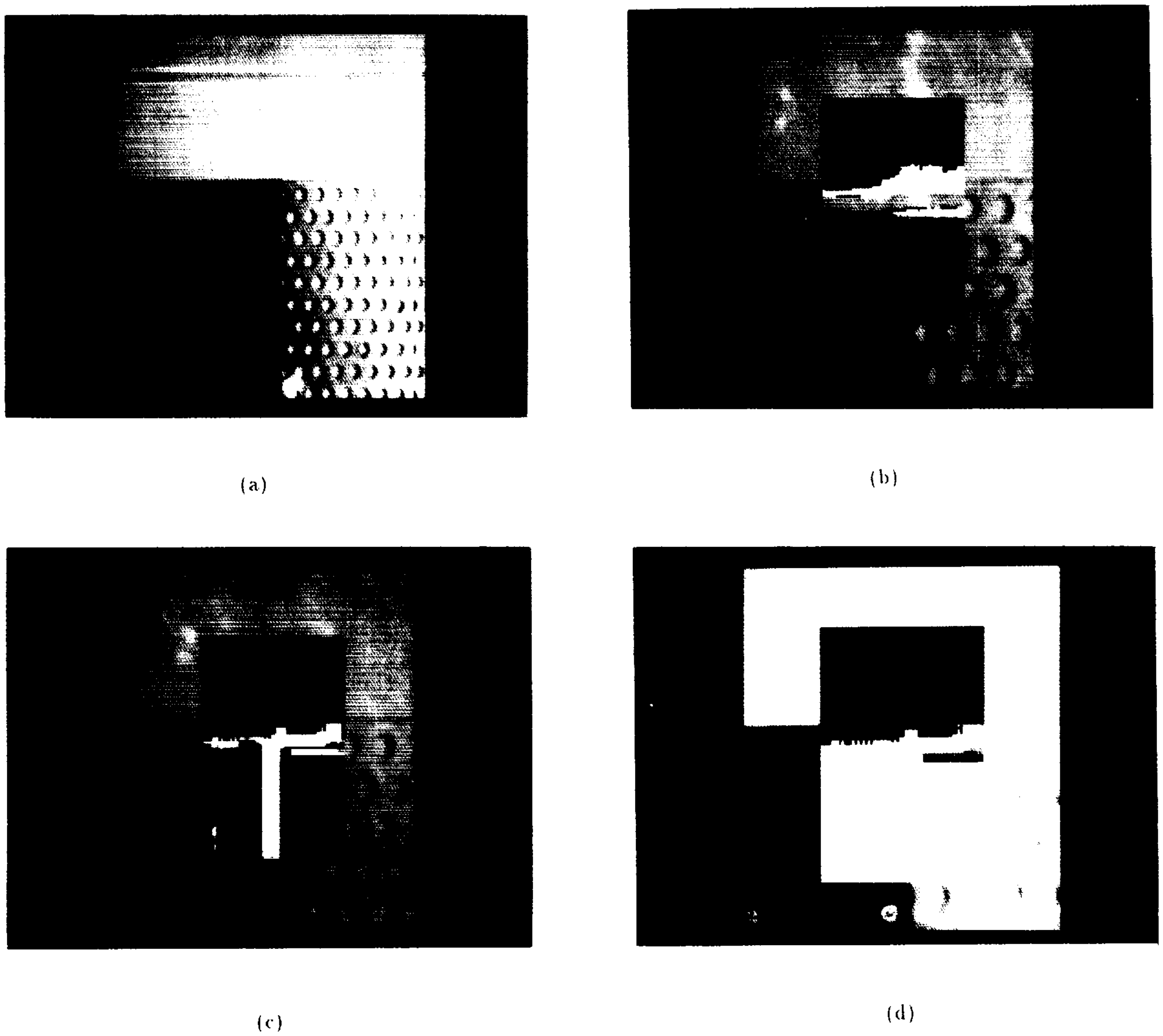


Fig. 2. (a) original, natural texture image, (b) segmentation by K-means algorithm, (c) energy segmentation by the improved ART2 only with noise reduction, (d) energy segmentation by the improved ART2 with both noise reduction and contrast enhencement.

a resonant state will be achieved quickly when that pattern is presented, adaptive process will reinforce the memory of the stored pattern by formulas. If the pattern is not immediately recognized, the network will rapidly search through its stored patterns looking for a match. If no match is found, it will enter a resonant state whereupon the pattern will be stored as a new category for the first time if unused neurons in the competitive layer exist. Otherwise, a new neuron should be created automatically by the autogrowing mechanism the F_2 layer to store the new pattern. Thus, the network is able to respond fastly to previously learned data, yet learn novel data when those are presented.

4. Experiment results

The performance of the segment algorithm by improved ART2 is examined by a series of experiments on image containing different textures. The size of each image is 100×100 with 256 gray levels. The size of maximum and minimum measuring window is defined as 11×11 and 33×33, respectively.

For the texture features from the image by fast SGLDM algorithm, the K-means algorithm is used [17] (shown in Fig. 2 (b)). However, the K-means algorithm has following disavantages:

- Supervised learning mode: the number of clusters must be set in advance. the different number may classify different results;
- Slow real-time ability: time of classification will raise at exponent rate with the cluster number increased;
- Unstability: the results of classification depends on the precision of feature extraction. when the extraction of the texture features has slightly change, the classifing result might be difference.

Compared to the K-means algorithm, the ART2 has many advantages, such as unsupervised training, high computation rates, and great degree of fault tolerance (stalility/plasticity).

In our test, the features, i.e. energy, entropy, correlation, homogeneity and inertia (or called as contrast), are used in texture analysis. The features have been proved to be a high degree of accuracy for the extraction of texture image features[3]. The parameters a, b, c, d, e, θ and ρ is selected in advance. a=b=10., c=0.25, d=0.8, e=10^{-6}. the selection of θ depends on different texture features and quantized angles of the features. For instance, the noise of each angle for the energy feature in the test is similar, so the value of θ is selected as 0.23 in every angle of the feature. On the other hand, the noise of each angle for the contrast feature is slightly different. the θ is set to 0.1, 0.12, 0.2, and 0.1 for the feature along to angle $0°$, $45°$, $90°$, and $135°$, respectively. The Fig 2. (a) is the original texture image. The Fig. 2. (c) is the segmenting result of the improved ART2 only with noise reduction. It is seen from the Fig. 2. (c) to greatly improve the segmentation of the texture image. The Fig. 2. (d) shows that the segmentation operation is further good after not only the noise reduction but also the feature enhencement are done.

5. Conclusion

The SGLDM provides the most powerful statistical representation for segmentation and identification of texture images. Its problem, consuming time has been improved greatly by a fast algorithm.

An improved *Adaptive Resonance Theory* (ART2) for analog input patterns is adapted to classify the image based on a set of texture image features extracted by a fast SGLDM. The non-linear thresholding functions in the ART2 F_1 layer have been composed of two parts: to reduce the effection of image noises on the features, a set of sigmoid functions is chosen depending on the types of the feature; to enhence the contrast of the features, we adopt *fuzzy multi-region mapping functions* The cluster number in output layer can be increased by an autogrowing mechanism constantly when a new pattern happens. The system written by C language is performed on a SUN-4/330 sparc-station with an image board IT-150 and a CCD camera.

6. Acknowledgements

The authors acknowledge the discussion made by A. Acciani and G. Bianco to provide us wiht C library.

References

[1] Van Gool L., P. Dewaele, and A. Oosterlinck (1985). Texture analysis anno 1983. *Comput. Vision Graphics Image Processing* 29, 336-357.

[2] Haralick R. M., K. Shanmugam, and I. Dinstein (1973). Texture features for image classification. *IEEE Trans. on Syst., Man, and Cybernet.* 3. 610-621.

[3] Weszka J. S., C. R. Dyer, and A. Rosenfeld (1976). A comparative study of texture measures for terrain classification. *IEEE Trans. on Syst., Mam, and Cybernet.* 6. 269-285.

[4] Lee J. H. and N. H. Lee (1992). A fast and adaptive method to estimate texture statistics by the spatial gray level dependence matrix (SGLDM) for texture image segmention. *Pettern Recog. Letters* 13. 291-303.

[5] Laws K. L. (1980). Rapid texture identification. *Proc. SPIE* 238. 376-380.

[6] Therrien C. W. (1983). An estimation-theoretic approach to terrian image segmentation. *Comput. Vision Graphics Image Processing* 22. 313-326.

[7] Cross G. and A. K. Jain (1983). Markov random field texture models. *IEEE Trans. Pattern Anal. Mach. Intellegence* 5. 25-39.

[8] Derin H. and W. S. Cole (1986). Segmentation of textured images using gibbs random fields. *Comput. Vision Graphics image processing* 35. 72-98.

[9] Derin H. and H. Elliott (1987). Modelling and segmentation of noising and textured images using Gibbs random fields. *IEEE Trans. Pattern Anal. Mach. Intellgence* 9. 39-55.

[10] Galloway M. M. (1975). Texture analysis using gray level run lengths. *Comput. Vision Graphics Image Processing* 4. 172-179.

[11] Sun C. and W. G. Wee (1983). Neighboring gray level dependence matrix for texture classfication. *Comput. Vision Graphics Image Processing* 23. 341-352.

[12] Carlton S. G. and O. R. Mitchell (1977). Image segmentation using texture and gray level. *IEEE Proc. Conf. on Pattern Recog. and Image Processing*.

[13] Mitchell O. R., C. R. Myers, and W. Byone (1979). A max-min measure for image texture analysis. *IEEE Trans. Comput.* 2. 408-414.

[14] Hsiao J. Y. and A. A. Sawchuk (1989). Unsupervised textured image segmentation using feature smoothing and probabilistic relaxation techniques. *Comput. Vision Graphics Image Processing*. 48. 1-21.

[15] Kundu A. and J. L. Chen (1992). Texture classification using GMF bank-based subband decomposition. *CVGIP: Graphical Models and Image Processing* 54. 369-384.

[16] Ohanian P.P and R. C. Dubes(1992). Performance evaluation for four classes of texture features. *Pattern Recognition* 25. 819-833.

[17] Tou J. T. and R. C. Gonzalez(1977). *Pattern Recognition principles*. Addison-Wesley Publishing Company, 377.

[18] Kandel A (1986). *Fuzzy mathematical techniques with applications* Addioson-Wesley Publishing Company, 274.

[19] Heermann D. P.(1992).*et al, IEEE Trans. on Geoscience and Remote Sensing* 30(1), 81.

[20] Freeman A.J. and D.M. Skapura(1992). *Neural networks, algorithms, application, and programming techniques*. Addison-Wesley Publishing Company.

[21] Hertz J. A. krogh, and R.G. Palmer(1991). *Introduction to the theory of neural computation*. Addison-Wesley Publishing Company.

[22] Rumelhart D.E., J. L. McClelland, and PDP Group(1988). *Parallel distributed processing, explorations in the microstructure of congnition, volume 1: foundations*, Cambridge: MIT press.

[23] Kohonen T.(1989). *Self-organization and associative menory*. SV, Berlin, New York.

[24] Carpenter G. A. adn S. Grossberg(1987). A massively paralle architecture for a self-organizing neural pattern recognition machine. *Comput. Vision, Graphics, Image Processing*. 27, 54-115.

[25] Carpenter G. A. and S. Grossberg(1987). ART2: Self-organization of stable category recognition codes for analog input patterns, *Appl. Opt.* 26, 4919-4930.

[26] Carpenter G. A. and S. Grossberg(1988). The ART of adaptive pattern resognition by a self-organizing neural network, *Computer*, 77-88.

[27] Carpenter G. A. adn S. Grossberg(1990). ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures, *Neural Networks*, 3, 129-152.

[28] Gan K.W. and K. T. Lua(1992). Chinese character classification using an adaptive resonance network. *Pattern Recog.* 25(8), 877-882.

[29] Tiraks A, L Sukissian and S. Kollias(1990). An adaptive technique for segmentation and classification of textured images, *INNC 90 Inter. Neural Network Conf.*, 1, 31-34.

[30] Lu S. and A. Szeto(1991). Improving edge measurement on noisy images by hierarchical neural networks, *Pattern Recog. Letters*, 12, 155-164.

[31] Keyvan S. and L. Rabelo (1992). Sensor signal analysis by neural networks for surveillance in nuclear reactors, *IEEE Trans. on Nuclear Science*, 39(2), 292-298.

[32] Wang Z. L.(1993). *IEEE Inter. Conf. on Fuzzy System*, California, March.

[33] Wang Z. L., G. Sylos Labini, and M. De Sario(1993).A self-organizing network of alterable competitive layer for pattern cluster,*ICANN Inter. Conf. on Artificial Neural Networks*. Amsterdam. The Netherlands. Seotember.

[34] Wang Z. L.(1993).A self-organizing network of alterable competitive layer for pattern cluster, *WIRN 93 6th Italian Workshop on Neural Nets*, Vietri sul Mare, Salerno, Italy, May.

[35] Wang Z. L. and M. De Sario(1993). Design principle for artificial neural network system, *Workshop on Neural Networks, Theory, Algorithms and Cognitive Modelling*, Bari, Italy, June.

→ MG
Finished
DRD

6C 45

AIAA-94-0887-CP

N94- 30550

MICROWAVE VISION FOR ROBOTS

Leon Lewandowski (603) 885-4281 and Keith Struckman (603) 885-5059
of Lockheed Sanders, Inc., Nashua, NH

7
p. 9

ARTIFICIAL NEURAL NETWORK – MICROWAVE VISION

Microwave Vision (MV), a concept originally developed in 1985 [1] could play a significant role in the solution to robotic vision problems. Originally our Microwave Vision concept was based on a pattern matching approach employing computer based stored replica correlation processing. Artificial Neural Network (ANN) processor technology offers an attractive alternative to the correlation processing approach, namely the ability to learn and to adapt to changing environments. This paper describes the Microwave Vision concept, some initial ANN-MV experiments, and the design of an ANN-MV system that has led to a second patent disclosure in the robotic vision field[2].

MICROWAVE VISION CONCEPT

Microwave Vision is similar to a bistatic radar system: Electromagnetic waves are radiated into the observation space, the reflected signals are received and processed to yield range and bearing to the object. Typically radars radiate pulsed RF signals. MV is instead based on the measurement and processing of a distinctive set of spectral lines. Similar to some high resolution radars, MV identifies the object by the spectral character of the reflected returns. MV differs from bistatic radar systems in two important aspects: 1) MV signals span much larger radio frequency bandwidths and 2) MV systems operate in the "near field" of the object. Precise position information and accurate object identification is achievable when operating at short ranges over very wide frequency ranges.

The spectra returned from different objects become more distinct by using an illumination spectrum that spans the natural electromagnetic resonance of these objects. For example, identification of a 10 cm tall object is based on signals containing frequencies in the neighborhood of 3 GHz. Figures 1, 2 and 3 demonstrate a simple version of the MV concept. One dipole transmits and the second receives a

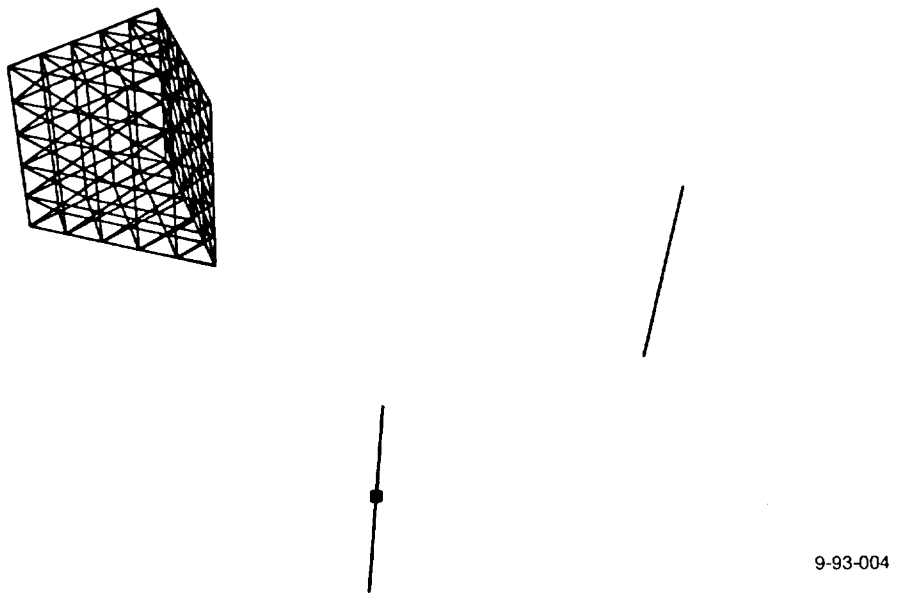


Figure 1. Six cm Tall Equiangular Wedge and Dipole Array Geometry

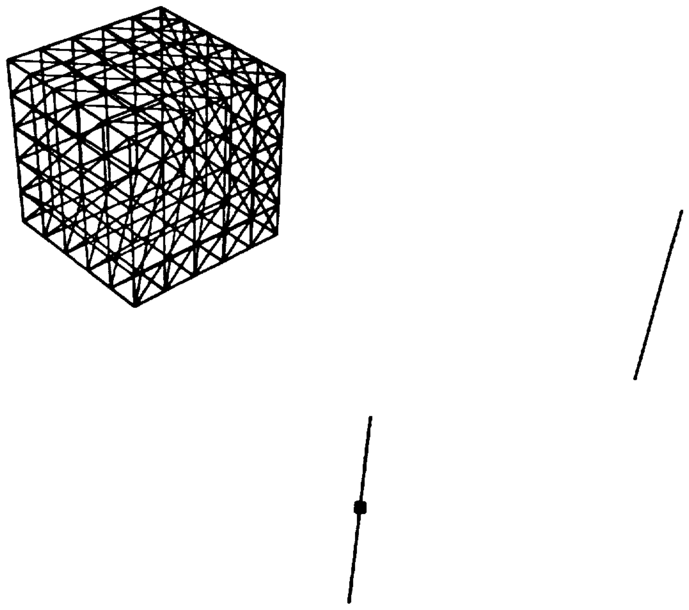


Figure 2. Six cm Cube and Dipole Array Geometry

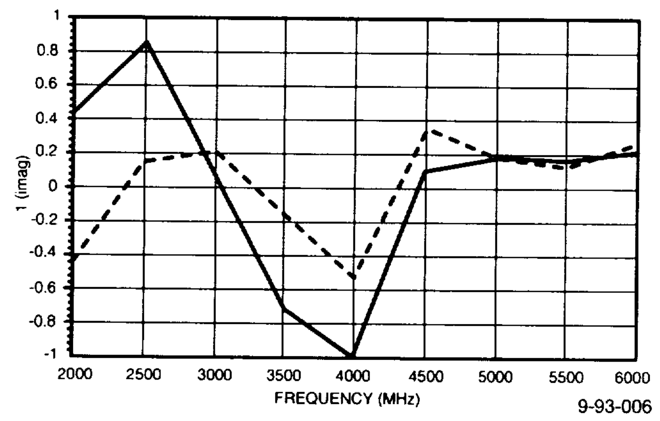


Figure 3(a). Receive Current Level (Imaginary Part) Cube (solid line), Wedge (dashed line)

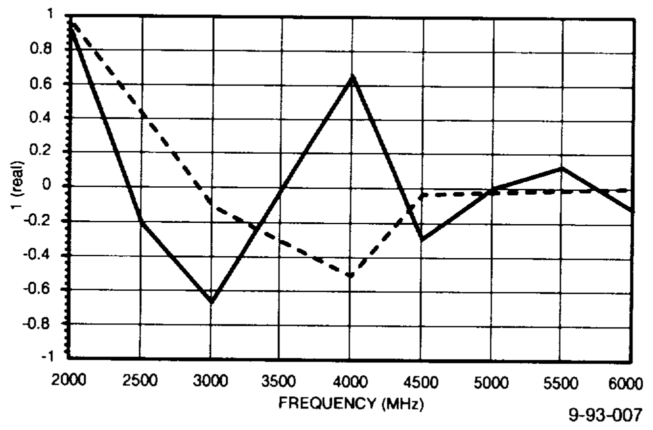


Figure 3(b). Receive Current Level (Real Part) Cube (solid line), Wedge (dashed line)

set of 10 spectral frequencies, evenly distributed between 2 GHz and 6 GHz. The reflected signals, as measured by the current on the second dipole, are strongly dependent on the particular object illuminated as shown by comparing the spectrums shown in Figure 3. Here, real and imaginary spectral components of the RF signal, reflected from the 6 cm tall equiangular wedge, and those reflected from the 6 cm cube are displayed. The two objects are clearly distinguishable through the contrast of their respective spectral returns.

The original MV concept was based on the correlation of measured spectral patterns with patterns "measured" from previous calibrations. During these early experiments the Correlation Coefficient R was recorded as a function of the water depth in a coffee cup. When the cup was full, the correlation to a previously recorded full cup spectral pattern was equal to unity. As the water depth was reduced, the spectrum responses changed, reducing the correlation value from unity to a minimum of 0.25 when the cup was completely empty. This simple experiment clearly demonstrated that the MV correlation process can yield information that is difficult to acquire with purely optical systems. The original correlation process was effective, but the trainable ANN processing technique has many additional advantages.

Artificial neural networks are ideal for use in an MV system, because unlike a computer or signal processor they are not programmed in the classical sense, but are instead trained using in this case, the MV spectrum measurements as the training stimulii.

ANN-MV PROOF OF CONCEPT SYSTEM

Our experimental ANN-MV system, shown in Figure 4, was trained to guide a simple robotic hand to a position that encloses the object. This system, used transmit and receive antennas mounted on the robotic hand to excite and receive reflected signals from simple objects. A center Vivaldi antenna sequentially transmitted a set of discrete signals that were received by the two outer antennas that form a pair of fingers on the robotic hand. Two sets of measurements are needed to resolve the signals reflected from the illuminated object. Each measurement set is recorded when the HP measurement channel is sequentially connected to one of the outer antennas. Object location measurements contain the sum of two sets of spectrums, the spectrum of the signal directly transmitted from antenna to antenna plus the desired signal spectrum that represents the signal radiated to the object of interest and reflected into an outer antenna. The reflected signal spectrum of interest is obtained by subtracting an initially measured baseline spectrum, a spectrum which was recorded when the object was absent. The resultant reflected signal is then inserted into the first layer of the ANN system.

Artificial neural network processing, as used in ANN-MV, is based on training the connecting weights between an input layer, a hidden layer and the output layer of an i80170 Intel Processor. Other ANN processing algorithms or processing techniques could have been investigated, but the availability of the Intel Chip and the relative ease of back propagation training [3] led to early experiments using the unit.

Many ANN based system applications are plagued with preprocessing problems associated with the generation of input vectors significant to

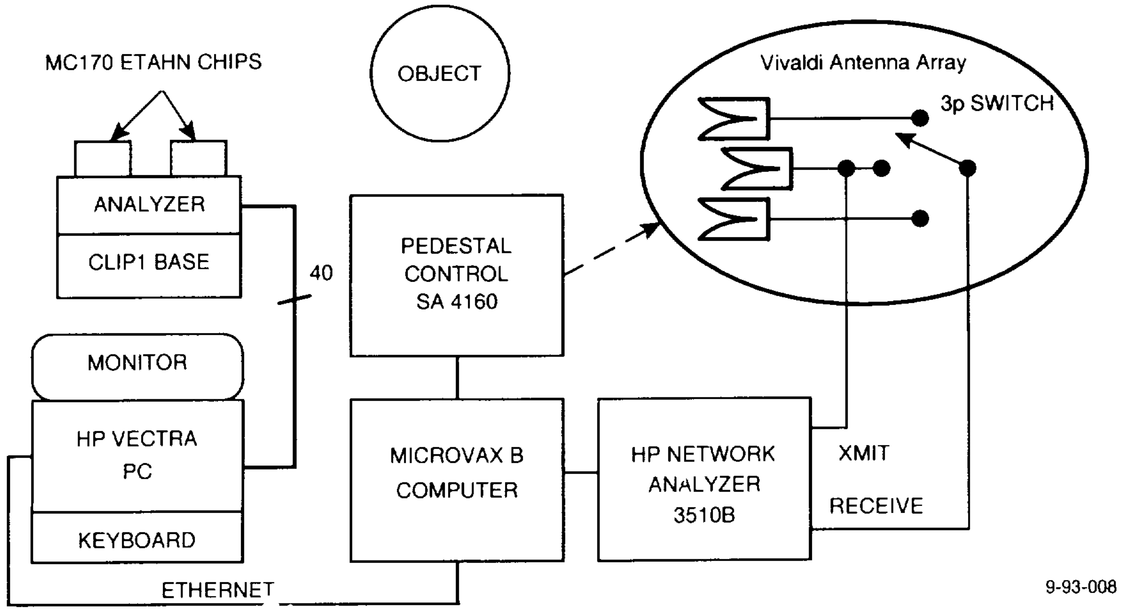


Figure 4. Schematic of Experimental ANN-MV System

the resolution problem. Microwave Vision affords a natural set of input vectors, i.e., those real and imaginary parts of the spectral lines reflected from the object and recorded at the receive antennas, as shown in Figure 3. As mentioned previously, the spectral lines should encompass the natural electromagnetic resonant frequency of the unknown objects.

The signals are weighted and summed at both the middle ANN layer and the output ANN layer. Rudimentary training process would be the task of forcing the output of an artificial output neuron (K) to be high when the input vectors correspond to reflections from object (K) but low for reflections from all other objects. This particular problem is a relatively easy ANN-MV task for many categories of objects. The robotic vision problem is significantly more complex since the robot needs to also measure the location and orientation of the object.

Our original ANN-MV task was to locate and move the hand to a soft drink can that was randomly located within a 50 cm radius/90° quadrant field of view. Most of the experiments were conducted by connecting the input layer containing 32 artificial neurons to middle layer consisting of 32 neurons and an output layer consisting of two neurons. The back propagation training algorithm was tasked to generate two outputs having patterns given by:

$$\bar{Y}_{out}(1) = \text{Range} * \text{Cos}(\theta)$$

$$\bar{Y}_{out}(2) = \text{Range} * \text{Sin}(\theta) \qquad \text{EQ-1}$$

Guidance to the hand was then given by a pair of simple calculations based on these two outputs. A complete set of input training vectors was obtained by sequentially positioning the can to 77 locations, every 15 degrees from -45 to +45 and 10 cm to 30 cm in 2 cm increments. At each location an (I) and a (Q) value was recorded for each of 16 frequencies between 2 GHz and 4 GHz. Exceedingly long, several hours, on chip training times were observed. Large robotic hand guidance errors were also measured unless the can was located very close to a training location. Subsequent tests showed that the input vectors changed markedly for small changes in can locations. These changes can be attributed to the phase rate of change with respect to centimeter changes in distance. At 3 GHz, a 2.5 cm range increase creates a two-way path change of 5 cm equivalent to 180 electrical degrees. This change dictates a training set based on differential ranges of approximate 0.5 cm.

Experiments with the initial ANN-MV processing technique demonstrated significant deficiencies in object location accuracies. These deficiencies were primarily caused by large input vector phase changes associated with distance changes normal to equal range contours, relative to the transmitter and receiver phase centers. This led to a system design that exploits "this" effect by sequentially preprocessing the input data as it is inserted into the ANN input layer. Initial investigations show that this preprocessing concept reduces the training time and sharply reduces residual training errors.

Object location algorithms are based on the intersection of equal time delay, elliptical contours. The transmit and right finger receive antenna are located at the foci of one set of elliptical contours, the transmit and left finger receive antenna are at the foci of the second set of elliptical contours. Figure 5 shows a pair of contours for two time delay paths from the center Vivaldi antenna to the Vivaldi antenna located on the right side of the hand. Each contour represents a particular time delay and therefore all object training positions along this contour can be operated on by the same set of phase unwrapping vectors. This phase unwrapping concept is the frequency equivalent of time domain range gating which is so effective in conventional radar systems.

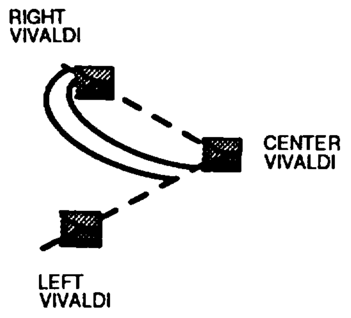


Figure 5. ANN-MV Elliptical Contours

Robotic control is based on two ANN Intel processors. The first processor receives inputs based on measurements between center antenna and the right finger antenna. Inputs to the second processor are based on center antenna to left finger antenna measurements. Each ANN processor performs identical operations which is to calculate and identify the contour having the highest probability of containing the object.

A set of 32 complex spectral responses are calculated by measuring the signal transmitted from the center antenna reflected from an unknown object and received by the antenna mounted on the right finger. As with the initial system, these spectral values are obtained by subtracting an object absent baseline spectrum from the total measured spectrum. The reflected spectral components are sequentially phase unwrapped and sequentially input to the first ANN layer. A unique set of phase unwrapping vectors are calculated for each contour within the object field. The exact number of independent contours is based on size of the field and the illumination frequencies.

Each object is represented by a set of output neurons which have previously been trained to identify the object and the location contour. Output neurons are observed as the first set of input vectors are sequentially unwrapped and input to the first ANN processor. The correct output neuron should go high when the input vectors are incremented to the delay associated with the contour containing the object.

The second ANN processor is served with its set measurement vectors and the outputs observed as the measurement vectors are unwrapped and input. Again, an output neuron should go high at the delay corresponding to the contour that intersects the object. The intersection of the two elliptical contours having high output states identifies the location of the object. One contour is calculated by the first ANN processor, the second contour is calculated by the second ANN processor.

Back-propagation training is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward perceptron and the desired output. This technique requires a differentiable function that is non-linear, which for the Intel i80170 chip is the conventional sigmoid function. The training of either of the processors, for a field containing a single object will be described. This training starts by initializing the ANN processor weights to small random values. The next step is to calculate the output of this processor using the spectrum values measured at the start of a contour and unwrapped for it's delay. The weights are adjusted to minimize the error, (output – desired output)^2 by a recursive algorithm that adjusts the weights by starting at the output nodes and working back through the hidden layer. This process is iterated through many cycles as spectrums recorded along all elliptical contours are sequentially input. The process is stopped when the residual is within predetermined acceptable limits. Figure 6 is a simplified sketch of the desired output function. The output neuron designed to identify the contour C(L) should be high for any of the unwrapped input spectrums recorded when the object was located on or near this particular contour. Connections shown in Figure 6 are limited to those connected to the first perceptron of the hidden layer.

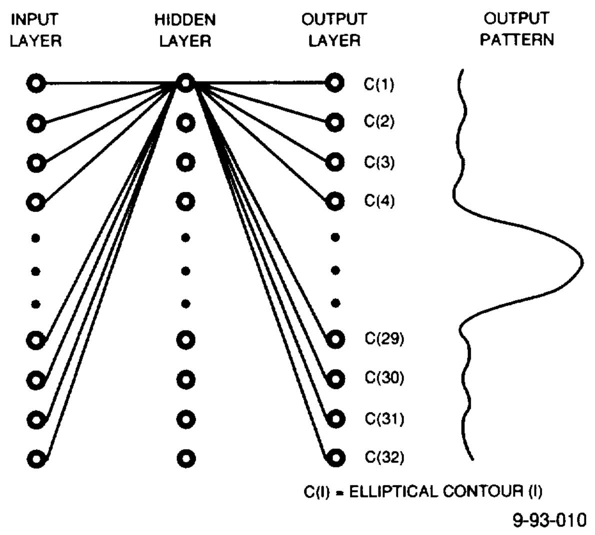


Figure 6. ANN Training Pattern

A modification to a probability based DF emitter location algorithm, is used to estimate the location of and object. Histories of all previous estimations provide increasingly accurate joint probability location estimations as additional measurements are performed.

A high neuron output representing a particular elliptical contour indicates that there is a high degree of probability that the object is on or near this contour. This probability is represented by a surface density that has unity height along the contour and has the conventional gaussian shaped pattern in directions normal to this surface.

Conventional radar range equations predict measurement accuracies that are inversely proportional to range to the fourth power. This range effect is included in our object location estimations by using standard deviations given by:

$$\sigma(r) = \sigma_{\text{min range}} \left[\frac{\text{range}}{\text{min range}} \right]^4 \quad \text{EQ-4}$$

This increase in sigma at longer ranges produces a probability surface that has a rapidly rising ridge in the direction normal to the contour containing the object when these contours are approached from the side nearest the robotic hand.

Conceptual probability surface densities generated for a cube located in front of robotic hand mounted array is shown below in Figures 7(a), 7(b) and 7(c). Figure 7(a) shows an unnormalized theoretical probability density surface based on the elliptical contours associated with the center-right antenna pair. This depiction demonstrates the start of the process used to locate an object, such as the cube shown in Figure 7(a). Figure 7(b) shows the surface associated with the center-left antenna pair. Figure 7(c) is joint probability density surface generated by the product of the surfaces shown in (a) and (b).

A short series of tests were conducted to verify the ANN-MV concept. The proof of concept was based on the second training and processing method. These tests used the experimental system shown in the schematic, Figure 4, to record process and move a robotic hand toward a simple object. The final goal of these experiments was to accurately move the robotic hand into a position that would permit the grasping of a small object. The robotic fingers on the simple hand was not moveable so this next step in the general solution to robotic problems could not be demonstrated.

Several key indicators, each pointing to successful experiments, were observed as the experimental process proceeded. The first of these was the ease of Intel i80170 ETANN chip training. The i80170 chip can be trained in two distinct ways. The slow direct way is to train with the chip-in-the-loop. We used a second faster way that records a typical on chip sigmoid function, then places this function into an external program that emulates the chip and trains with a procedure identified as off-line learning.

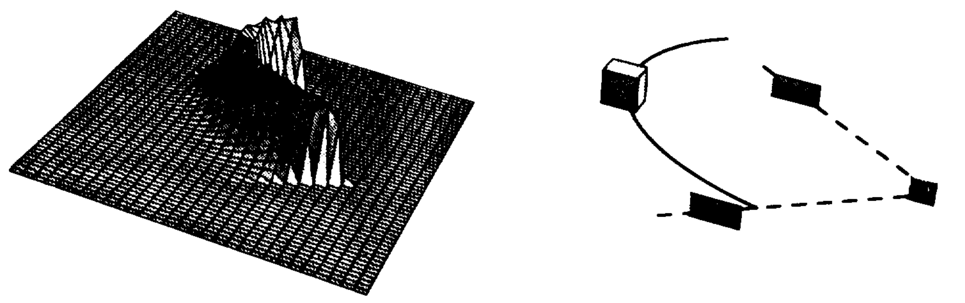

Probability density surface
Right elliptical equal time delay contour

Figure 7(a). Surface and Contour Based on Center to Right Antenna Measurements

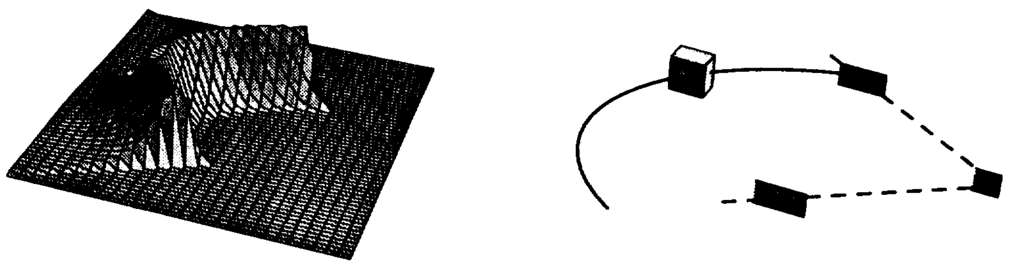

Probability density surface
Left elliptical equal time delay contour

Figure 7(b). Surface and Contour Based on Center to Left Antenna Measurements

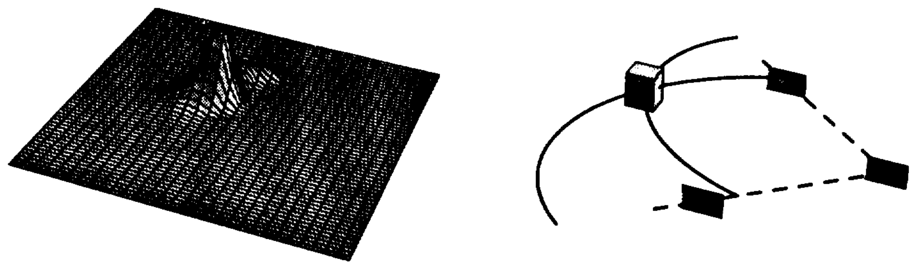

Joint probability density surface
Left/Right elliptical equal time delay contours

Figure 7(c). Surface and Contours Based on Previous Two Sets of Measurements

An alternate is to learn off-line, then download these neuron weights, and then follow with the more accurate chip-in-the-loop learning. The off-line learning process produced accurate guidance commands when used in conjunction with our second unwrapped vector input technique. Chip-in-the-loop training was not required. A strong indicator of robust robotic operation was the ability of the hand to follow a can that was moved between processing steps.

A HP 8510 network analyzer was used to measure the reflected signals at sixteen uniformly spaced frequencies between 2 GHz and 6 GHz. Probability density surfaces were computed by the Vectra PC using outputs generated by the two ANN chips. The maximum of the product of these surfaces identifies the location of the object, which for this set of experiments was the coordinates of an aluminum soda can. Figure 8 shows the product probability estimate based on calculations generated as the robotic hand progressed from its (0., 0.) starting location. The final pair of contours were based on artificial neural network output processed microwave spectrums recorded at a hand location of 7.3 cm, 18.1 cm). The sharp peak at (8 cm, 28 cm) is within approximately 2 cm of the correct location. When the robotic hand moves to this location, it is in very close to the desired location. Subsequent moves of an articulated hand could accurately close on this cylindrical object.

CONCLUSIONS

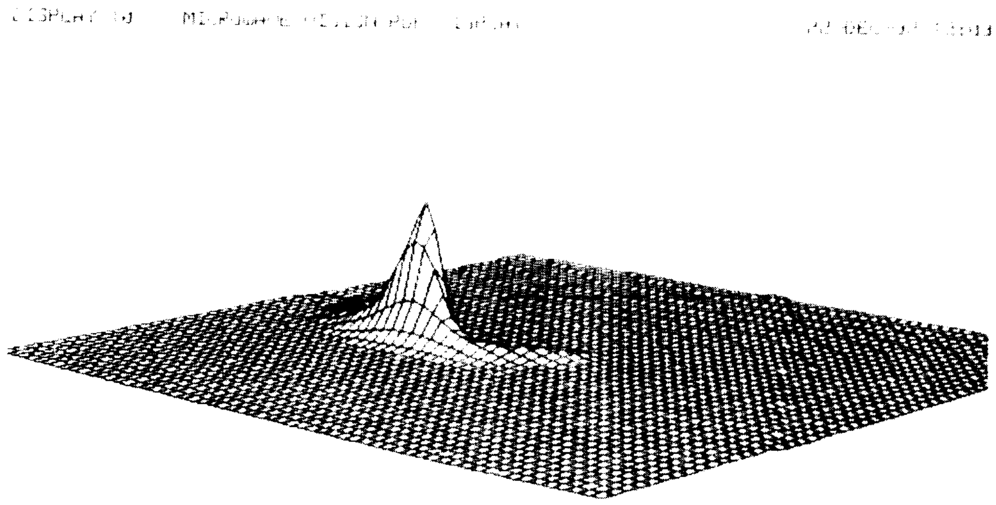
The techniques describe herein provide the first stage in the solution to many robotic vision problems. The next stage, that of providing objects coordinates and subsequent movements for grasping, a difficult problem for optical vision systems, should be a fairly simple problem for Microwave Vision-Artificial Neural Network processing. Here, the robots fingers are in the electrical near field of the object where increasingly accurate microwave measurements can be performed. The Range^4 problem no longer applies. At this point the elliptical contour technique will be discarded and it is anticipated that full cross spectrum ANN training commands will be applied. In the simplest sense, as the antennas on the robotic fingers approach the object, their radiation will be blocked, generating a clear signal that the fingers are ready to touch the object. Obviously the MV-ANN system will not look for this condition, instead the ANN processor will have been trained to output a signal that indicates that the hand has "CLOSED ON THE OBJECT".

REFERENCES

[1] K.A. Struckman, "Microwave Vision for Robots", Patent Disclosure D-4030, Sept. 18, 1985.

[2] K.A. Struckman and R. Martel, "Microwave Vision for Robots Using Artificial Neural Network Processing", Dec. 22, 1992.

[3] P. J. Werbos,"Backpropagation through time: what it does and how to do it", Proc. IEEE, pp, 1550-1560, Oct. 1990.



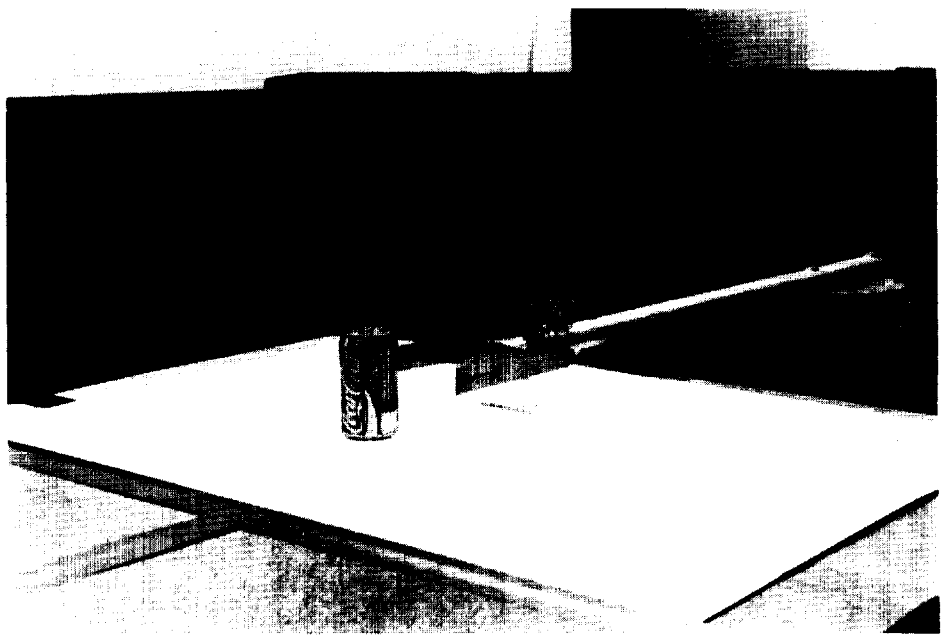


Figure 8. This is the Final ANN-MV probability Density Surface Calculated for this Experiment. Here the Object, a Soda Can is Still at (10 cm, 28 cm), but the Hand has Moved Forward to Coordinates (7.3 cm, 18.1 cm). This Last Joint Probability Surface Maximum has a well Defined Peak at (8 cm, 28 cm).

AIAA-94-1200-CP

N94-30551

AN ELECTROMAGNETIC NONCONTACTING SENSOR FOR THICKNESS MEASUREMENT IN A DISPERSIVE MEDIA

S25-63

6046

[illegible]

p. 5

Robert L. Chufo
Electrical Engineer
U.S. Bureau of Mines
Pittsburgh, Pennsylvania

Abstract

This paper describes a general purpose imaging technology developed by the U.S. Bureau of Mines (USBM) that, when fully implemented, will solve the general problem of "seeing into the earth." A first-generation radar coal thickness sensor, the RCTS-1, has been developed and field-tested in both underground and highwall mines. The noncontacting electromagnetic technique uses spatial modulation created by moving a simple sensor antenna in a direction along each axes to be measured while the complex reflection coefficient is measured at multiple frequencies over a two-to-one bandwidth. The antenna motion imparts spatial modulation to the data that enables signal processing to solve the problems of media, target and antenna dispersion. Knowledge of the dielectric constant of the media is not necessary because the electrical properties of the media are determined automatically along with the distance to the target and thickness of each layer of the target. The sensor was developed as a navigation guidance sensor to accurately detect the coal/noncoal interface required for the USBM computer-assisted mining machine program. Other mining applications include the location of rock fractures, water-filled voids, and abandoned gas wells. These hazards can be detected in advance of the mining operation. This initiating technology is being expanded into a full three-dimensional (3-D) imaging system that will have applications in both the underground and surface environment.

Introduction

Early research investigated various high-frequency radar sensor systems using pulse, impulse, FM-CW, or synthetic pulse. Electromagnetic waves penetrate coal, rock, and earth, but when the energy penetrates the media, the returning information content appears to be scrambled and out-of-focus. The problem is dispersion: Media dispersion, coupled with antenna and target dispersion, cause problems too complex to analyze in the time domain. These problems are much easier to resolve in the frequency domain. Both the time domain and frequency domain are transforms of only one variable, so either approach is legitimate. However, it is very difficult to work problems in both domains at the same time. The theory supporting the present research is in the frequency domain, but the resulting architecture for signal processing uses both. The concept being used is the spatial-domain technique (i.e., moving the antenna to create a modulation on the radar output). This concept has been applied to advantage to reject unwanted reflections and help cancel out media dispersion and antenna dispersion.

The problems of designing an underground imaging sensor were solved by utilizing a sensor model created from one-dimensional, spherical wave, scattering matrix theory. By separating surface reflections from single-layered media reflections, laboratory and field testing confirmed the validity of the one-dimensional imaging model. The model, based on fundamentals, allows the use of a wide range of design architectures.

Rather than devoting project time to hardware development, emphasis was placed on the data processing scheme required to derive coal thickness and dielectric constant from network analyzer measurements of the reflection coefficient of the target media. The sensor data were taken at a wide range of frequencies and antenna positions (e.g., 401 frequencies between 600 and 1,600 MHz and 32 equally-spaced positions over a distance of 16 in). This was accomplished with a vector network analyzer connected to an antenna that was moved in space by a linear positioner. Data processing provided a direct measurement of the thickness of underground media and also the electrical characteristics of the media. Prior knowledge of the characteristics of the media is not necessary.

Measurements made of coal, rock, concrete, granite, and salt have shown that the technique can measure thickness from 0 to over 5 ft in single and multilayer media. The accuracy of the technique is not affected when the material is rough or wet. These results and parallel applications such as the measurement of the depth of hidden tunnels, the thickness of multilayer highway pavement, and the location of buried nuclear waste, unexploded ordnance, and cultural artifacts, have provided the technical incentive to further develop this unique technology to take advantage of its broad potential, including full 3-D imaging of the underground environment.

Acknowledgment

The author acknowledges the contributions of Walter J. Johnson, WALTECH Holography for the development of the algorithms used in the radar coal thickness sensor and Raymond J. Helinski, USBM, for constructing the spatial modulator and assisting in data collection during both surface and underground testing.

Experimental Measurement Technique

The original development of the radar coal thickness sensor measurement technique started with Kerns plane wave theory [1] and solved for each of the scattering coefficients in the matrix with a standard antenna calibration technique. However, when the antenna was within two wavelengths of the material being measured, diffraction became too strong an effect for a plane wave model. A solution was sought in simple plane wave theory but the cost was the necessity of an explicit solution for the coefficients of the model. Several approximation techniques for linear calibration were tried and it was found that a solution was possible. The coefficients did not vary when the antenna was closer than two wavelengths to a metal calibration plate. This was called the linear reduction method [2] and it worked quite well except for some second-order problems. It was assumed that these problems were nonlinear multipath effects that were not accounted for in the calibration procedure. The math model was then expanded to include the higher order effects and named the "quick reduction method." Many of the higher order coefficients were lumped together to accommodate a metal plate calibration technique preformed with a 4-ft-square metal plate placed between the sample and the antenna. This in-situ calibration technique corrected for signals returned from reflectors beyond the edges of the calibration plate. In an underground mine these reflections would be from material similar to the target coal but outside the measurement area. Presently a self-calibrating technique is being evaluated that will improve upon the metal plate calibration technique.

The thickness measurement process begins with a measurement of the input reflection coefficient of an antenna in close proximity to the coal surface. This measurement is taken at a wide range of frequencies and positions (e.g., 401 frequencies between 600 and 1,400 MHz, and 32 positions between 4 and 20 in from the coal. This is accomplished using a vector network analyzer connected to an antenna moved in space by a linear positioner. The measurement plane is then electrically moved from the instrument measurement plane to the plane of the antenna. Figure 1 shows the instrumentation setup.

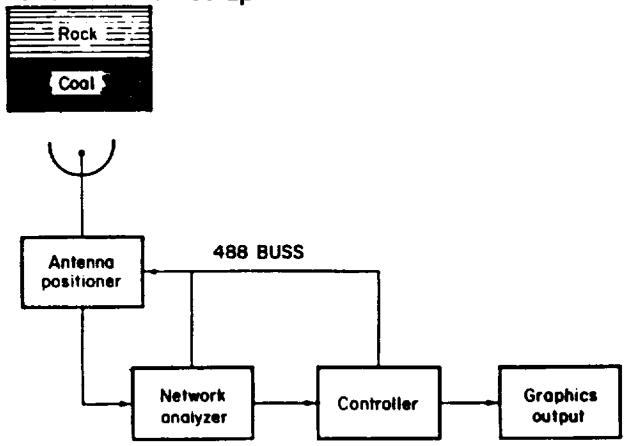


Fig. 1. Instrumentation

Data

The data from this measurement are a function of both frequency and position. The data contain both amplitude and phase information. Transforming the data to the time domain at this point in the process and inspecting the time domain history for this one antenna position shows the absence of any sharp peaks around the antenna, indicating that the information for the coal surface is corrupted by other effects such as the antenna dispersion, diffraction, and multipath. These effects must be characterized and accounted for by considering the frequency domain history at each antenna position.

Antenna Transfer Functions

To characterize the antenna, a separate test is run with a metal surface substituted for the coal surface; the same frequencies and positions are used. This provides data from a known reflection surface to obtain the antenna transfer functions. These functions are used in removing antenna dispersion from the data taken at the corresponding antenna position.

Removing Antenna Dispersion

When the antenna transfer functions are accounted for in the data, the result is the product of the antenna-to-surface-to-antenna distance, represented by the spatial delay and the coal surface reflection coefficient. It is the reflection coefficient that contains the information for the coal thickness. Other reflections (i.e., multipath) are also present in the resulting transfer function.

Shifting Image Plane to Coal and Removing Diffraction

Dividing the reflection coefficient expression by the spatial delay shifts the image plane from the antenna to the surface of the coal and removes diffraction. With the antenna-to-surface-to-antenna distance a known quantity, the true reflection coefficient in the frequency domain can be determined.

Integrating Space

By performing a spatial integration, the multipath can now be decorrelated. Since spatial integration is coherent with the coal but not coherent with any other spatial distances, the multipath will become zero sum or at least small compared with the surface reflection.

Transforming to the Time Domain and Range Gating

Transforming the data to the time domain and range gating, removes unwanted reflections from the data by gating out all the information on either side of the desired peak. However, the data as presented in the results section of this paper have shown that range gating is unnecessary as the signal return from the dielectric interfaces between the coal and rock are sufficiently distinct to make the interface easily discernible. If range gating were used, transforming the reflection coefficient back to the frequency domain would yield the composite reflection coefficient for just the reflections within the range gate.

Validation of the Model

An earlier method [2] for determining the thickness transformed the time domain reflection coefficient back to the frequency domain so that the measured reflection coefficient could be correlated with theoretical reflection coefficients for various thicknesses and dielectric constants. The theoretical reflection coefficient that correlated best, provided a statistical determination of the thickness and dielectric constant of the coal being measured. For example, the theoretical reflection coefficient that correlated best with the measured reflection coefficient was for a coal thickness of 5.3 in (for a relative dielectric 4 and loss tan of .03). The actual thickness of the rough wet coal measured in this underground test was a nominal 6 in. This result provided the encouragement to refine the model and proceed with the development of a method to directly measure the coal thickness from data acquired by the network analyzer frequency domain measurements.

Field Test Results

The purpose of this research was to develop a coal and rock thickness sensor of sufficient accuracy to provide vertical and horizontal guidance of both room-and-pillar and highwall mining machines. In order to validate the theory developed for thickness measurement, extensive underground and surface mine testing was performed. Over a period of 2 years, tests were conducted in mines with a variety of geological and environmental conditions. Test areas of both freshly mined and aged coal from 3 to 60 in thick were measured. The areas measured ranged from very dry to extremely wet with water dripping from the roof test area. The wet coal did not affect the thickness measurement. Coal seams with clay and metal vein intrusions of iron pyrite could be imaged and the distance from the coal surface to the intrusion was accurately measured. Surface roughness and cleating was not a problem. The average thickness of rough cleated surfaces was measured accurately. Accurate measurements were obtained even when water filled the cracks between the cleats.

Roof Tests

Roof thickness tests were made in production mines and in the Safety Research Coal Mine at the USBM Pittsburgh Research Center. Figure 2 is a representative measurement of roof coal thickness. On the vertical axis, the plot shows the amplitude of the reflected signal in decibels; time in nanoseconds is shown on the horizontal axis. The large peak on the vertical axes represents the reflection from the first interface, the air/coal interface. Signals plotted to the left of the large peak represent discontinuities internal to the measurement equipment and between the antenna and the coal surface. These reflections are reduced to at least 30 dB below the air/coal reflection by the calibration and spatial integration scheme. To the right of the air/coal reflection are reflections from discontinuities internal to the coal and shale being measured. The printout on the left is the thickness of coal between the air/coal interface and the coal/shale interface. Measurements have identified both the thickness of the coal and the thickness of the next layer, usually shale, above the coal roof. At the L-band frequencies presently used, the depth of penetration is usually about 10 ft. Future roof thickness measurement research will attempt to provide a direct readout of the thickness of each layer of geological material within the penetration range of the signal. At the present time, the power level of the transmitted signal is 0 dBm (1 mW). This signal level, or less, is adequate to produce a good signal-to-noise ratio for the return signal measurement. The hardware will permit an increase in transmitted power of 20 dB to determine if greater penetration is best achieved through increased signal power or through the use of a lower transmitter frequency. Both the hardware and software will operate from 300 kHz to 3,000 MHz.

Rib Tests

Figure 3 is a plot of actual data taken at an operating highwall mine in West Virginia.

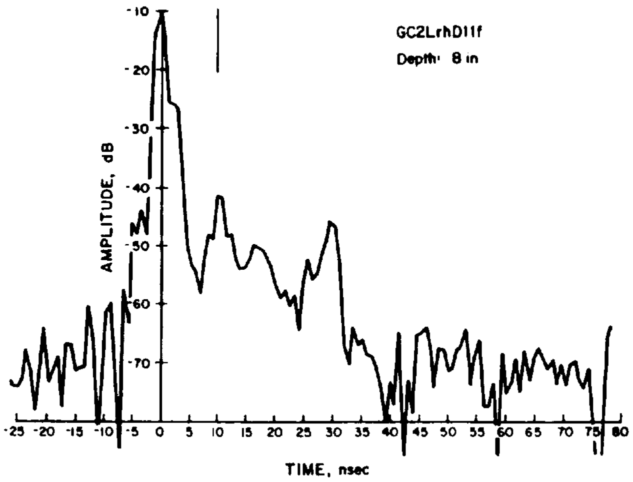


Fig. 2. Roof Coal Thickness

Measurements were made in freshly mined entries immediately following the mining machine. The determination of rib thickness can usually be interpreted both visually from the FFT data in the figure and from the numerical readout from the automatic thickness measurement software. A large dielectric contrast is seen at the first air/coal interface and a somewhat smaller but still pronounced reflection can be seen as the signal exits from the coal rib at the coal/air interface in the adjacent drift. The vertical bar to the right of the main peak at the first interface as the signal enters the rib indicates that the rib thickness is 35 in. The dielectric constant and loss in decibels per meter is also indicated above the rib thickness measurement printout at the left of the plot. Rib measurements were also made in underground mines over the range of 18 to 50 in. The thickness in these test ribs could be determined to within 1 or 2 in.

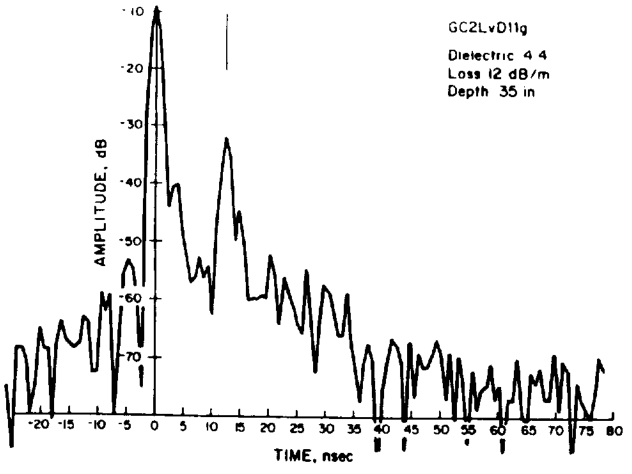


Fig. 3. Rib Thickness

Figure 4 is a plot of actual data taken at an operating highwall mine in Kentucky. In this case the software presented an amplitude vs range plot, an improvement over the amplitude vs time plot of figure 3. The data shown are for a rib thickness of 57.2 in with a dielectric constant of 3.97. The measured loss was 2.92 dB per wavelength. Also measured but not shown was the distance from the antenna measurement plane to the coal surface. This distance data could be used for control of the position of the mining machine.

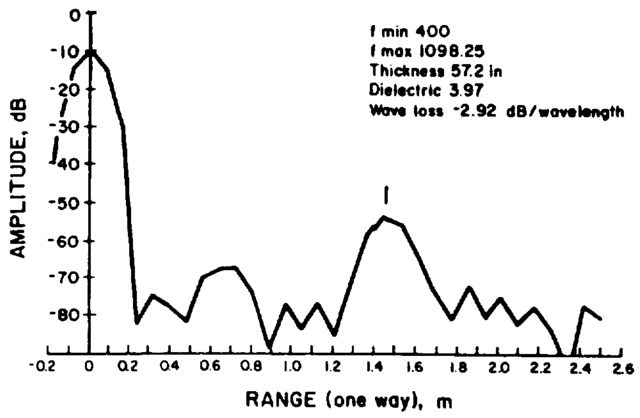


Fig. 4. Rib Thickness, Amplitude vs. Range

Thickness Measurement of Other Materials

The thickness of other materials has been measured with equal success. Granite, sandstone, and concrete ranging in thickness from 2 ft to over 4 ft have been measured to within 2% of their actual thickness. The thickness of each layer of multilayer pavement can be determined as can the location and orientation of steel reinforcing bars.

Conclusions and Recommendations

Field testing of the electromagnetic coal thickness sensor has produced results of sufficient accuracy (1 in for coal from 3 to 60 in thick) to justify continuing with the engineering work necessary to develop a practical sensor that can be mounted on a mining machine for the determination of roof, floor, and rib thickness. In addition, this research will be extended to the development of a full 3-D imaging system capable of "seeing into" the earth. Algorithms are presently being evaluated to simultaneously measure the azimuth, elevation, and range of targets in multilayered media such as coal and rock as well as for the location of buried ordnance and nuclear waste. Future plans are to minimize the size of the data set to reduce the software processing time now about 1 sec, and facilitate the construction of a compact sensor suitable for machine mounting or use as a general geological survey tool.

It was found that vertical E-field polarization penetrates thicker coal ribs than horizontal E-fields.

This is thought to be due to the thin horizontal ash layers having a higher loss than the coal.

The real part of the dielectric in coal varies very little from a value of 4 but the loss tangent varies a great deal. Wet, rough, or heavily cleated coal had little effect on the dielectric measurement.

The ash content may be related to the loss tangent of the dielectric measurement. This would be a helpful means to identify higher quality coal.

References

1. Kerns, D. M., Plane-Wave Scattering-Matrix Theory of Antennas and Antenna-Antenna Interactions, Nat. Bur. Stand., NBS Monogram 162, June 1981.

2. Chufo, R. L and Walter J. Johnson, A Radar Coal Thickness Sensor, The 1991 IEEE Industry Applications Society Annual Meeting, September 28-October 4, 1991, Dearborn, MI.

AIAA-94-1201-CP

PERCEPTION SYSTEM AND FUNCTIONS FOR AUTONOMOUS NAVIGATION IN A NATURAL ENVIRONMENT

N94- 30552

Raja Chatila, Michel Devy, Simon Lacroix, Matthieu Herrb
LAAS-CNRS
7, avenue du Colonel-Roche
31077 Toulouse Cedex - France
E-Mail : (raja|michel|simon|matthieu@laas.fr)

6047 P.

Abstract

This paper presents the approach, algorithms and processes we developed for the perception system of a cross-country autonomous robot. After a presentation of the tele-programming context we favor for intervention robots, we introduce an adaptive navigation approach, well suited for the characteristics of complex natural environments. This approach lead us to develop an heterogeneous perception system that manages several different terrain representations. The perception functionalities required during navigation are listed, along with the corresponding representations we consider. The main perception processes we developed are presented. They are integrated within an on-board control architecture we developed. First results of an ambitious experiment currently lead at LAAS are then presented.

1 Context - Introduction

A large amount of results exists today on mobile robot navigation, most of them related to indoor environments. As for outdoor navigation, most of the works concern environments wherein obstacles are rather structured, and the terrain mostly flat (*e.g.* road following [1]). More recently, studies considering autonomous mobility in *natural unstructured* outdoor environments comes out [2] : several applications are considered, such as public safety [3] (fire fighting, chemical disaster...), sub-sea intervention or exploration, and planetary exploration [4, 5].

Several aspects make these kinds of interventions a demanding and difficult problem for robotics :

• The robot has to operate in a natural, unstructured, maybe hostile and a priori unknown environment ;

• There might be interaction discontinuities with the robot because of communication breakdowns, important delays or low bandwidth ;

• The information on the robot and the environment is mostly acquired through the robot's own sensors.

These constraints rule out direct teleoperation as well as telerobotics approaches, and point towards robots with **important autonomous capacities** : the environment being poorly known and the communication possibilities very poor, the mission can only be predefined at a *task-level* in general, not in its every details. The robot must then build and maintain its own representations of the environment, upon which it autonomously reasons and plans the actions to perform in order to fulfill the mission.

As opposed to behavior-based control schemes [6], we favor the development of a global architecture with two main parts to tackle this challenge [7, 2] : an operating station for mission programming and supervision, and a remote robot system[1] able to *interpret* the mission and *execute* it autonomously.

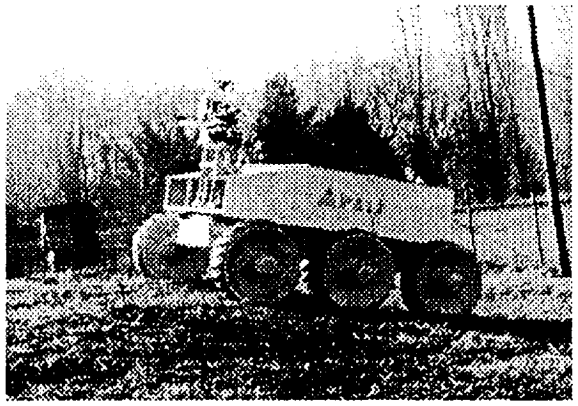


Figure 1: **The mobile robot ADAM in its environment**

The operating station includes the necessary functions that allow a human to (i) build an *executable robotic*

[1]not necessarily a single one.

mission that can be interpreted and executed by the robot, (as opposed to a higher level description of objectives) ; and to **(ii)** supervise its execution, taking into account the delays and communication constraints. Its presence essentially ensues from the following considerations :

• The mission is not defined once and for all : according to returned data, one should be able to change the objectives of the mission (when unexpected events occur for instance) or to decide the execution of a particular action (such as "pick this sample" in the case of a scientific exploration).

• The robot could fall into difficult situations wherein its own capacities are insufficient, a human intervention would then be necessary for troubleshooting.

As for the robot, its autonomy essentially relies on its ability to build faithful representations of its environment, which is obviously necessary for him to interpret the mission and decompose it into executable tasks, considering its actual context.

We focus in this paper on the development and organization of the perception functionalities an autonomous cross-country robot must be embedded with. The following section introduces the general adaptive approach we chose to tackle with outdoor environment navigation, that emphasizes the need to develop several perception processes. Section 3 presents the different perception functionalities required during navigations, and the corresponding terrain representations maintained by the robot. The processes we developed to build these representations are presented in section 4, and the way they are controlled and integrated within the context of our robot architecture is presented in section 5. We finally describe the first results of the EDEN experiment, currently developed at LAAS with the mobile robot ADAM[2] (figure 1).

2 A Multi-Purpose Perception System for Adaptive Navigation

The complexity of outdoor natural environments comes essentially from their diversity and lack of structure : some areas can be totally flat (maybe cluttered with easily detectable obstacles - big rocks lying on a prairie for instance), whereas others area can be much more cluttered, such as a landscape of smooth hills (sand dunes) or an uneven rocky area. This variety induces several different behaviors, and constrains both the perception and motion planning processes.

According to a general economy of means principle (on-board processing capacities, memory and time are always limited), we favor an *adaptive* approach [8, 9] : we aim at adapting the robot behavior of the robot to the nature of the terrain, and hence three navigation modes are considered :

• And a **reflex** navigation mode : on large flat and lightly cluttered zones, the robot locomotion commands are determined on the basis of *(i)* a goal and *(ii)* the information provided by "obstacle detector" sensors.

• A **2D planned** navigation mode : it relies on the execution of a planned 2D trajectory, using a binary description of the environment in terms of *Crossable/Non-Crossable* areas.

• A **3D planned** navigation mode : this mode requires a precise model of the terrain, on which a fine 3D trajectory is planned and executed.

Each of these navigation mode is suitable for a particular terrain configuration, and requires a specific representation. Besides this trajectory planning functionalities, there are some other important processes that also require a representation of the terrain : exteroceptive localization, often required to refine or correct the estimation of the robot position provided by its internal sensors ; and *navigation planning*, which is in charge of intermediate goal and navigation mode selection.

Several authors emphasized on the development of perception and motion planning processes able to deal with any terrain configuration [10, 11], trying to recover as much information as possible from the acquired 3D data. Besides the processing complexity, such an approach has a main drawback : it does not takes advantage of the variety of the environment. Although sometimes needed, the recovery of a complete and accurate 3D geometrical model may be often not necessary : more simple and approximative representations will be sufficient in many situations, when the terrain is mostly flat for instance.

We believe that aiming at building such a "universal" terrain model is extremely difficult and not efficient, and we therefore chose to endow the robot with a *multi-level* terrain modeling capacity : a particular representation is built or updated only when required by a given task. This involves the development of various perception processes, each of them being dedicated to the extraction of specific representations (*multi-purpose perception*).

At each step of the incremental execution of its mission, the navigation planner autonomously chooses an intermediate goal, along with the navigation mode to apply to reach it. This induces the choice of the representations it must update, which comes to answering these questions : which sensor to use ? With what operating modalities ? How should the data be processed ? Perception planning becomes in our case a

[2]ADAM : Advanced Demonstrator for Autonomy and Mobility, is property of Framatome and Matra Marconi Space, currently lent to LAAS.

key component to enhance the robot autonomy and efficiency.

To achieve this, we propose to build and update systematically a *global qualitative* description of the environment on which all "strategic" decisions are taken. This representation is built thanks to a fast analysis of the raw 3D data acquired (either by a Laser Range Finder - LRF - or by a stereovision correlation algorithm), that provides a terrain description in term of navigation classes, and some other *qualitative* informations, such as the possible presence of a landmark, the mean altitude and slope of some areas... Each time this representation is updated, it is structured in order to produce a semantically significant model, from which navigation and perception plans are deduced.

3 Terrain Representations

After a brief presentation of the perception functionalities and the constraints brought by outdoor environments, we introduce in this section a *multi-level* environment model, that defines the relations between the various representations.

3.1 Outdoor Representations : characteristics and constraints

The difficulty of representing outdoor environments comes essentially from the fact that they are not intrinsically structured, as compared to indoor environments where simple geometric primitives match the reality. As a consequence, any representation based on geometric primitives (linear or second degree surfaces, super-quadrics...) is difficult to build and to maintain, and introduces an approximation of the reality via artificial structures. We therefore favored the development of simpler representations (polygonal maps, elevation maps...), easier to build and manage. Semantic informations are not explicitly contained in such representations, but can anyhow easily be extracted.

The other characteristics of the representations are related to the robot sensors and mission :

• The sensors are always imperfect : their data are incomplete (lack of information concerning existing features) and not precise. They generate artifacts (information on non-existing features) and errors (wrong information concerning existing features). The same area when perceived again can therefore be differently represented. Hence environment representations must tolerate important **variations** [12].

• The environment is initially unknown (or very poorly known) and is **incrementally discovered** : the robot must be able to manage local momentary representations, and merge them in global descriptions of the world. We are convinced that global representations are required [13], especially to recover from deadlocks that often appears when dealing only with local representations.

Finally, one must not forget that the system memory is limited, and so the representations must be as compact as possible.

3.2 Perception Functionalities and Corresponding Representations

3.2.1 Trajectory Planning

From the poorest to the richest, here are the representations required by the three navigation modes we retained :

• **Reflex Navigation** : The robot locomotion commands are determined on the basis of *(i)* a target value (heading or position) and *(ii)* the information provided by "obstacle detector" sensors. An obstacle avoidance procedure enables the robot to move safely, and the area to cross is essentially obstacle-free, so that there are poor chances that the robot fall into deadlocks. Strictly speaking, this mode does not requires any modeling of the terrain, but a description (a simple 2D polygon in our case) of a zone where it can be applied.

• **2D planned navigation** : This mode is applied on lightly cluttered environments, that can be represented by a binary description in term of *Crossable / Non-Crossable* areas. The crossable zones are the places where the robot attitude is not constrained, *ie.* where the terrain is mostly flat, or has an admissible slope for the robot to run safely, whatever its heading position is. A trajectory defined by a sequence of 2D positions is planned within the crossable areas. In our case, the 2D planner requires a binary *bitmap* description, on which a distance propagation method (similar to those presented in [14]) produces a Voronoi diagram.

• **3D planned navigation** : On uneven or highly cluttered areas, the "obstacle" notion is closely linked with the constraints on the robot attitude, and therefore constrains the robot heading position. Planning a trajectory on such areas is a much more difficult task [15] that requires a detailed modeling of the terrain. In our case, the 3D planner builds its own data structure on the basis of an elevation map, computed on a regular Cartesian grid (section 4.4).

3.2.2 Localization

The internal localization sensors of the robot (odometry, inclinometers, inertial platform...) generate cumulative errors, especially on uneven or slippery areas. A localization procedure based on exteroceptive sensors is often necessary for both the robot and the supervising operator : to plan safe trajectories on formerly perceived areas for instance, the robot obviously needs to know precisely where it stands ; and a false position value may mislead the operator.

Such a localization procedure requires a specific *global* representation of the environment, be it a set of 3D points in the case of a correlation-based localization (iconic matching [16]), or a global map of detected landmarks (that must then be modeled, using particular geometric descriptions) in the case of a feature-based localization [17]. These two kinds of representations can be viewed as maps of interesting zones for the purpose of localization. In our case, we developed an original localization procedure (section 4.5), that requires a B-Spline based model of the terrain.

We are also currently investigating the modeling of unstructured objects (rocks, bushes...) thanks to complex geometric primitives (super-quadrics [18]) : such a model could be used to perform landmark detection, and might provide a "qualitative" localization functionality, sufficient in reflex navigation mode.

3.2.3 Navigation Planning

Navigation planning consists essentially in the determination of an intermediate goal, as well as the mode to activate to reach it, considering the mission's objective and the partial (and unprecise) knowledge the robot has on its environment. Several different constraints can be taken into account to perform this "route" planning, depending on the context : one may prefer execute safe trajectories from the localization point of view, or one may choose the fastest trajectories (time constraint), the shortest (energy constraint)... A semantic significant description of the perceived environment is here necessary. We have chosen a topological connection graph (section 5.2.2) : such a structure can contain very rich informations, and a theoretical formalism, often applied in the robotic community [19], is available for its exploitation.

3.2.4 Perception Planning

Perception planning, which is closely linked to navigation planning, requires a prediction ability : given a sensor and a point of view, what can be perceived ? To answer this question, the perceptual constraints of the sensor (occlusion, field of view, specularity) must be checked considering an environment numerical model.

3.3 A Structural Scheme

Several data structures that represent the same entities in the environment must coexist in the system. In this *multi-layered heterogeneous model*, the different representations are easily managed and a global consistency can be maintained. The relationships between the various representations explicit their building rules, and defines a constructive dependency graph between them. The figure 2 illustrates these relationships : each thin arrow represents a data processing algorithm, and the thick straight arrows corresponds to the production of a structure required to a trajectory planner. We distinguish two kinds of dependencies :

• **Systematic** dependencies : Every time a representation is updated, all the representations that systematically depends on it (arrows labeled "S") are updated. As one can see on the figure, every time 3D data are acquired, the global bitmap representation, the region representation and the connection graph are updated. Let's also note that when a localization model is available, the informations it contains are merged in the connection graph (section 5.2.2).

• **Controlled** dependencies (labeled "C") : The representations that are not always necessary are only built under control of the navigation planner. For instance, an elevation is only required to cross an uneven zone.

The top level of this heterogeneous model is a "bitmap" description of the environment, built upon the results of the fast terrain analysis algorithm. A lot of information is available in every pixel of this bitmap, such as the terrain label and its confidence level, the estimated elevation, the identification of the region it belongs to... We have chosen such a structure for the following reasons : it is simple, rich, adapted to the lack of geometrical structure of the environment and to the Digital Elevation Map description (section 4.4), and flexible, in the sense that any supplementary information can easily be encoded in a pixel without reconfiguring the entire description and the algorithms that use it. Moreover, the techniques that allow to extract structured informations (regions, connexity...) from a bitmap are well known and easily implemented.

3.4 Memory Management

The main drawback of maintaining global representations is memory occupancy, that rapidly becomes huge if they covers large areas, especially when using bitmap representations and elevation maps. To cope with this, we are currently developing a "forgetting" functionality : the area surrounding the robot, with a size limited by the sensor capacities, is fully described, whereas the remaining already perceived terrain is structured in a more compact way. The key point here is to determine the informations one must not forget : for the purpose of long range navigation, we consider that only the connection graph and the localization model are necessary to maintain.

We consider two different ways to implement this : the first one is to take advantage of the global bitmap region structuration, or of any other classical data compression method. The precise informations brought by the possibly computed elevation maps is then totally lost. The second way is to use the B-Spline based representation : the B-Spline representation would then be systematically built (in parallel with trajectory execution for instance). Only the B-Spline representation,

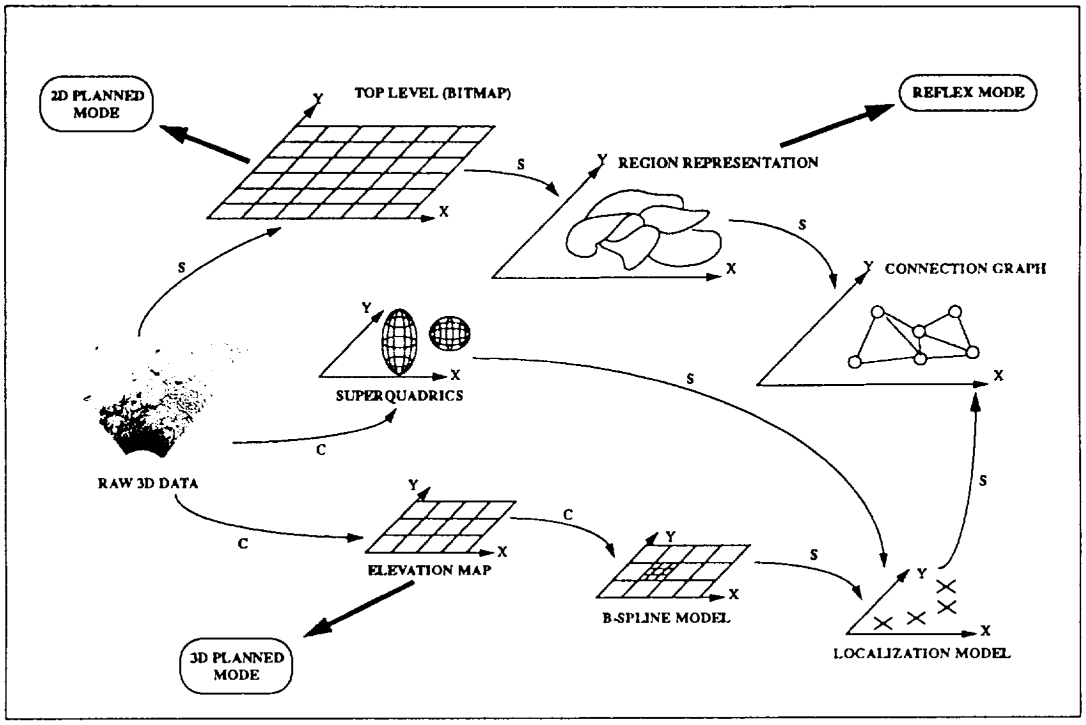


Figure 2: The representations used in the system

which is extremely compact, and that contains much more informations than the global bitmap representation, is kept in memory.

4 Building Representations

4.1 Fast Classification

Applied each time 3D data are acquired, this process produces a description of the perceived areas in term in *terrain classes*, along with some qualitative informations. It relies on a specific discretization of the perceived area in "cells", on which different characteristics that allow to label them are computed [9].
The discretization is the projection of a regular grid defined in the sensor frame (fig. 3). Its main characteristics are that it respects the sensor resolution, and that it points out a "density" attribute : the number of points of point contained in a cell, compared with a *nominal density* defined by the discretization rates, provides a useful information concerning the area covered by the cell : for instance, it is equal to the nominal density if the cell corresponds to a flat area. This information, along with other attributes concerning the cells (mean altitude, variance on the altitude , mean normal vector and corresponding variances) allows to heuristically label each cell as one of {*Flat, Slope, Uneven, Obstacle, Unknown*}.
This classification procedure, which complexity is $O(n)$, where n is the number of 3D points considered,

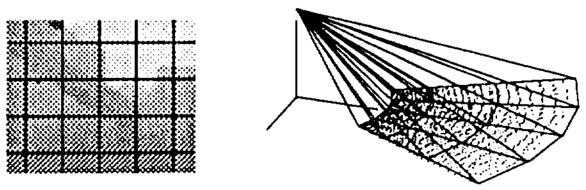


Figure 3: Discretization in the sensor frame, and projection on the ground

takes around half a second on a Sparc-10 workstation to process a 10.000-points 3D image. It has proved its robustness on a large number of different images (fig. 4), produced either by the LRF or a stereovision correlation algorithm[3], and is especially weakly affected by the sensor noise (uncertainties and errors). An important point is that it is possible to estimate a confidence value on the labeling of each cell : this value generally decreases with the distance of the cell to the sensor, because of the decreasing accuracy on a 3D point coordinates with this distance. But this confidence also obviously depends on the label itself : for instance, a flat cell containing a few erroneous points can be labeled as an "uneven" one, whereas the probability that erroneous points perceived on an actu-

[3]The discretization then differs slightly from the one used for LRF images

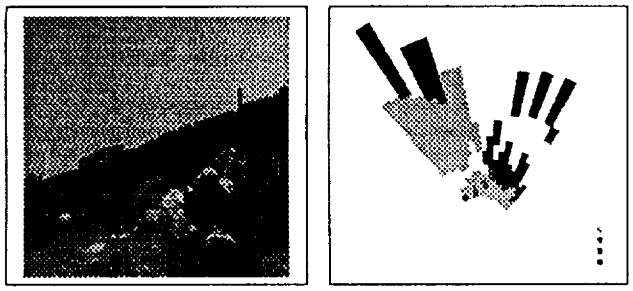


Figure 4: **Classification result on a complex scene. From clear to dark : Unknown, Flat, Slope, Uneven, Obstacle**

ally uneven zone lead to a "flat" label is very low. The quantitative estimations of this confidence value $P(error) = F(distance, label)$ are statistically determined, and constitute the useful model of the logical sensor "terrain classifier" (figure 5).

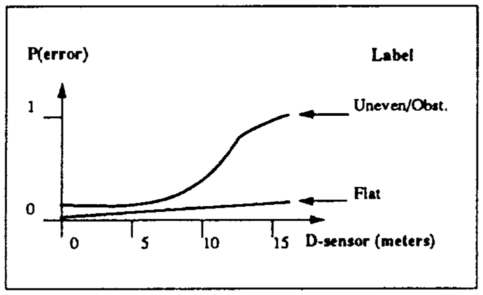


Figure 5: **Error probability on the cell labeling**

We are considering the application of a similar classification method on luminance images : global information concerning the same cells in the camera frame (color, texture...) should permit a fast determination of the terrain nature, and therefore produce a more significant description of the terrain. Another interesting thing to consider is the detection of areas of interest for the localization procedure (possible presence of landmarks or particular geometric features), using the attributes determined for each cell.

4.2 Global Model Building

In the incrementally built bitmap structure that represents the global terrain model, all the informations provided by the classification are encoded (label and corresponding confidence, elevation, slope). Fusion of the classifier output is a simple and fast procedure : each cell is written in the bitmap using a polygon filling algorithm. When a pixel has already been perceived, the possible conflict with the new perception is solved by comparing the label confidence values. This process is illustrated in figure 6 : the area originally labeled "obstacle" in front of the first position (left image) is split into two smaller obstacle areas plus a flat area when perceived from a smaller distance (right image). Many experiments have proved the robustness of this fusion method.

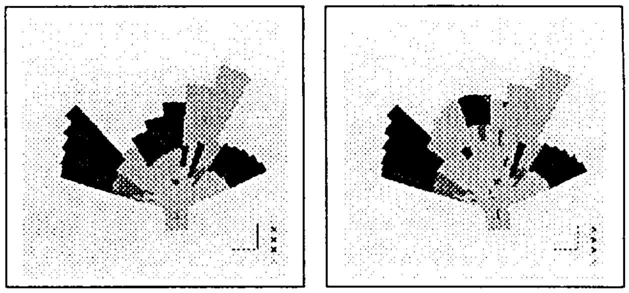


Figure 6: **Two steps of the global bitmap model building**

4.3 Connection Graph Building

Once the global bitmap representation is updated, it is structured in a "region model", thanks to classical image processing algorithms. Regions are areas of uniform label, uniform mean altitude and uniform confidence. If no precise geometrical informations are available in the description of a region, some useful qualitative informations can anyway easily be extracted, such as its surface or its including rectangle. A contour following algorithm provides all the neighborhood informations between the regions, that defines the topological connection graph. A node of the graph is related to the border between two regions, whereas an arc corresponds to the crossing of a region. Section 5.2.2 presents different ways to valuate the graph, considering the regions' attributes.

4.4 Fine Modeling

When an uneven area has to be crossed, it must be precisely modeled in order to plan a secure trajectory. We use for that purpose a generic interpolation method [20] that builds a discrete representation $z = f(x, y)$ on a regular Cartesian grid from a 3D spherical image $(\rho, \theta, \phi) = f(i, j)$.

Local Elevation Map (LEM) Building

Our method relies on the analysis of all sets of four neighboring points in the spherical image : they define *patches* in the Cartesian robot's redressed frame. Thanks to the fine grid resolution, a planar approximation is sufficient to represent a patch. The interpolation problem is then reduced to finding the intersection between each (x, y) "vertical" line and the plane that best approximate the patch. A test based on depth discontinuities allows to decide whether a patch can be interpolated or not, and leads to an estimation of the elevation Z_{Local} for the (x, y) interpolated points. An accuracy on each computed elevation is estimated, using Jacobian matrix of the sensor model to estimate variances on the raw Cartesian measurements, and a Kalman Filter to compute variances on the plane parameters [21].

Global Elevation Map (GEM) Building

A fusion of different LEM in a global elevation map may be needed for trajectory planning if the uneven area can not be entirely perceived from a single viewpoint. Once the estimation of the new robot's position is achieved (section 4.5), we combine the new LEM and the former Global Elevation Map into a new global map. The new elevation $(Z_{Global})_k$ after the k^{th} acquisition is updated by this ponderation equation :

$$(Z_{Global})_k = \frac{\sigma_{Z_G}^{-2}.(Z_{Global})_{k-1} + \sigma_{Z_L}^{-2}.(Z_{Local})_k}{\sigma_{Z_G}^{-2} + \sigma_{Z_L}^{-2}}$$

4.5 Localization Processes

Besides a localization process based on structured features [17], we developed a localization process that relies on a peak detection method [22], better suited for unstructured environments.

The specific terrain representation used here is a B-Spline surface based model, built upon an elevation map thanks to a least-square approximation. Such a model is very rich and compact, and provides a hierarchical description of the environment : a coarse level B-Spline representation is first computed on a uniform mesh, and a test based on the least-square errors points out the areas where some refinement is needed. A new mesh with smaller size patches is then defined, and a new B-Spline representation is computed, which ultimately leads to a tree model, in which each node corresponds to a B-Spline surface.

This analytic model allows to extract features such as high curvature points, valleys or ridges. We currently only implemented a peak extraction procedure based on a quick analysis of the *matrix* expression of the B-Spline surfaces. Once the peaks are extracted, we apply a feature matching localization method, cooperating with an iconic one : the iconic method is only performed in the neighborhood of the detected features. Hence, using small correlation windows, we avoid the long processing time usually encountered with such methods.

5 System Architecture and Control

The generic control architecture for the autonomous mobile robots developed at LAAS is organized into three levels [23, 24]. It is instantiated in the case of the EDEN experiment as shown in figure 7. The higher task planning level plans the mission specified by the operator in terms of tasks, with temporal constraints, executable by the robot. This operating station level, not currently used in the experiment, will be implemented in an specific environment to validate our teleprogramming approach.

Let's describe here the functional and decisional levels, and the way they are integrated.

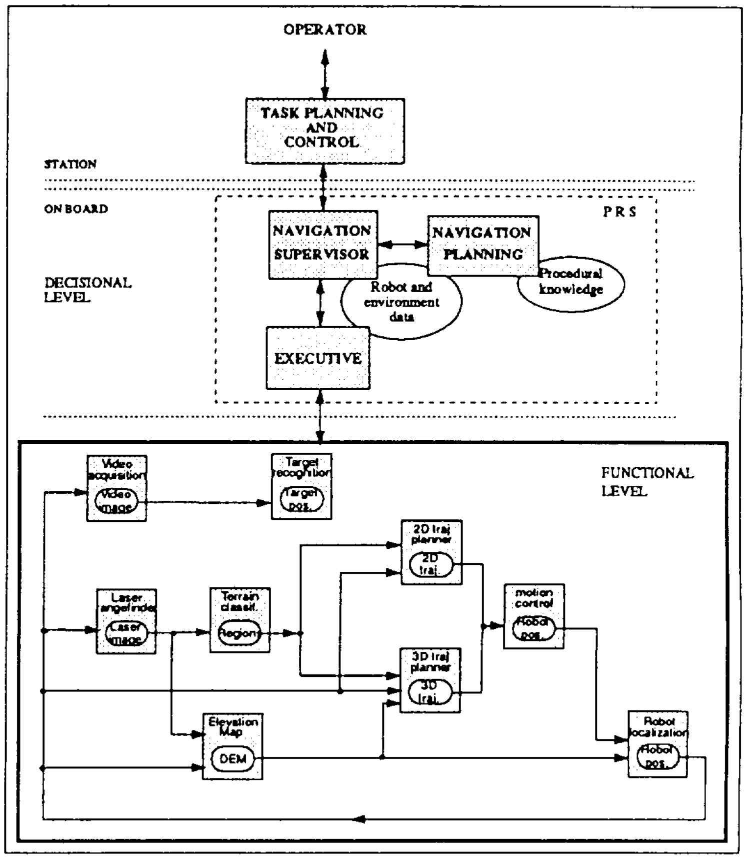


Figure 7: Global control architecture. Connections between the modules at the functional level show data flow.

5.1 The Functional Level

The Functional Level includes the functions for acting (wheels, perception platform), sensing (laser, cameras, odometry and inertial platform) and for various data processing (feedback control, image processing, terrain representation, trajectory computation, ...). To control robot functionalities and underlying resources, all these functions are embedded into modules defined in a systematic and formal way, according to data or resources sharing. Thus, modules are servers which are called via a standard interface, and allow to combine or to redesign easily the functions [25]. These modules can be viewed as a generalization of the logical sensor concept [26].

Figure 7 shows the set of modules used for the experimentation and the data flow during the progress of an iteration. The connections are dynamically established by the decisional level according to the context.

5.2 The Decisional Level

This level includes the navigation planner and a supervisor that establishes at run-time the dependencies between modules. It also controls their execution according to the context and the robot state, and installs the conditions/reactions in case of external events (watching for obstacles when executing a trajectory for instance). In our current implementation, the three entities of the decisional level have been simplified and merged together, using a Procedural Reasoning System [27].

5.2.1 The Supervisor and the Executive

The supervisor receives the task to be executed, described in terms of actions to be carried out and modalities. If the task is not directly executable (typically when the goal lies in an unknown area), the navigation planner refines it (section 5.2.2). The supervisor watches for events (obstacles, time-out, etc.) and reacts to them as planned, according to the dynamics of the situation and the state of the other running tasks. It sends to the Executive the different sequences of actions which correspond to the task, and sends back to the operator the informations related to task (*e.g.* specific data, and the report about its execution,etc.). The executive launches the execution of actions by sending the related requests to the functional level modules. It manages the access to resources and the coherence of multiple requests at the syntactic level, and can take into account the parallelism of some sequences (watching for obstacles while moving toward an intermediate goal for instance). It sends back to the supervisor reports about the fulfillment of those basic actions.

5.2.2 Navigation Planning

Generally speaking, the navigation planner uses procedures to carry out the task and decompose it into executable elementary actions, on the basis of the current environment and robot states. It is a key component of the decisional level : mixing both procedural knowledge and knowledge about the environment, it perform the decisions that provide the robot with a "smart" behavior. These decisions include *perception strategies*, *ie* the choice and the definition of the different perception tasks to perform, and *motion strategies*, that imply the definition of intermediate goals and the choice of navigation modes. The two problems are obviously closely linked, but to avoid a great complexity, we developed two independent techniques coupled afterwards.

Motion Strategies

The basic technique to plan a route in the known environment relies on the execution of an A^*-like search in the connection graph. This search selects a path, *i.e.* a succession of connected regions, that defines the intermediate goal and the motion mode to activate. The valuation of the arcs (that connect the region borders) is obviously determinant to implement different strategies. Our valuation is currently a heuristic mix between these criteria :

• **Arc label** : to plan a route that minimizes the execution time, the region label are taken into account. The planner then avoids to cross uneven areas when possible, since they require a fine modeling and a complex trajectory planning.

• **Arc confidence** : considering only the former constraint, the artifacts raised by the classification procedure (essentially badly labeled "obstacle" cells) would mislead the robot navigation. The arc label criterion is therefore pondered by its confidence, which allows the planner to cross some obstacles areas for instance, which actually triggers the execution of a new perception task when facing such areas.

• **Altitude variation** : For the purpose of energy saving, one may wish to minimize the positive altitude variations during trajectory execution, which increases the cost of climbing hills for instance.

Finally, let's note that a *localization ability* value can be taken into account while planning a route : from the localization model and the global bitmap model, landmarks (or interesting areas) visibility zones can be quickly computed, which produces a structure similar to a potential field. A localization ability value is then associated to each node of the graph, and a path that maximizes the sum of these values along the route can be determined.

Using some pre-defined rules, an analysis of the search result is then performed to define the next *perceptual need* among the three following : localization, discovery (perception of unknown area), and model refining (re-classification of an already perceived zone from a closest point of view or fine modeling).

Perception Strategies

Once the intermediate goal and the perceptual need are defined, the next perception task is performed according the following procedure [28] :

1. Perceptual **constraint checking** : characteristics on the sensor (field of view, resolution) and on the environment (visibility) constrains the observation ;

2. **Prediction** of the result of the perceptual task, *i.e.* estimation of the information it can provide ;

3. And finally **evaluation** of the contribution of the predicted task, in the context of the current need.

The main point here is to faithfully model the logical sensor to use ("classifier", "peak extractor",...), as in section 4.1.

As an example, let's examine a perceptual task selection : suppose the search in the graph derived a need to enhance the confidence value of a certain area. From the intermediate goal selected, the following procedure is run :

1. For each pixel of the global bitmap surrounding the sensor (within the LRF distance limit), the visibility constraint is checked using the elevation value encoded in the pixel ;

2. The current confidence label (Equal to zero if the pixel has not yet been perceived) of each perceivable pixel is compared to a theoretical "mean confidence value" the sensor can bring (deduced from the curves of figure 5). This comparison permits to estimate the amount of information the sensor can provide.

3. Finally, the usefulness of the predicted task is estimated, and the consideration of other constraints (allowed time, maximal sensor field of view...) defines its parameters, *ie.* perception direction, the LRF scanning mode, the field of view...

6 The EDEN Experiment

All the concepts and processes described in this paper are currently being integrated in the context of the "EDEN" experiment.

6.1 Experimental Test Bed

ADAM[4] has six motorized non directional wheels with passive suspensions, and is equipped with a "perception head" composed of a 3D scanning laser range finder with a deflecting mirror and two color cameras, mounted on a 1-axis pan platform.

The on-board computing architecture is composed of two VME racks running under the real time operating system VxWorks. They are connected to the operating station (a Sun SparcStation 10-41) by an Ethernet link. The first rack includes two 68030 CPUs and various I/O boards, and is dedicated to internal localization (thanks to odometry encoders and a inertial platform) and locomotion

The second rack is composed of two 68040 CPUS, three Datacube boards and some I/O. It is dedicated to sensing activities : video image acquisition, laser range finder command and acquisition, local processing of data.

During the experiments, most of the "high level" computing processes are run on the operating workstation to take benefit of a better debugging environment and of the pre-existence of the softwares under Unix. However, we have the possibility to embark all the softwares in a near future : some are already ported under VxWorks, and it is possible to use an on-board Sparc CPU under Sun-OS.

[4]Its chassis was built by VNII Transmach (S^t Petersburg, Russia)

6.2 Experiments

Figure 8: **ADAM's natural environment**

The figure 8 shows an illustrative image of ADAM's natural environment; it is a 20 by 50 meters area, composed of flat, sloppy, uneven rocky areas, and of big obstacle rocks. The canonical task is "GoTo Landmark", the environment being initially *totally unknown*. The goal landmark is currently a 2D pattern detected and localized in a luminance image. We have performed several "reach that goal" experiments using only the 2D motion planner in the crossable zones, and a "discovery" strategy. After a few "perceive - analyze - plan" steps, (from 3 to 10, depending on the chosen path) Adam reaches the target located at an approximatively 30 meter distance from its starting point. The whole processing time does not exceed half a minute at each step, but due to the slow motion of the robot (its maximum speed is 28 cm/s) and the LRF image acquisition time, ADAM takes generally about 15 minutes to execute its mission.

We have also performed experiments using only the 3D motion planner; for this sake, we have partially integrated the following functions : fine terrain modeling, localization procedures and 3D trajectory planning on uneven terrain[5]

Figure 9 illustrates the position update and the terrain model updating performed after the third acquisition : the left figure shows the extracted features in the Local Elevation Map, built from the third depth image ; the right figure presents the corresponding correlated points (and the correlation windows) in the current Global Elevation Map. Figure 10 represents the new Global Elevation Map after the robot position updating and the fusion.

[5]The computation time needed on a sparc II Sun station to build a Digital Elevation Map is about 2 sec.; the localization process takes about 3 sec., and the 3D planning process needs about 60 sec.

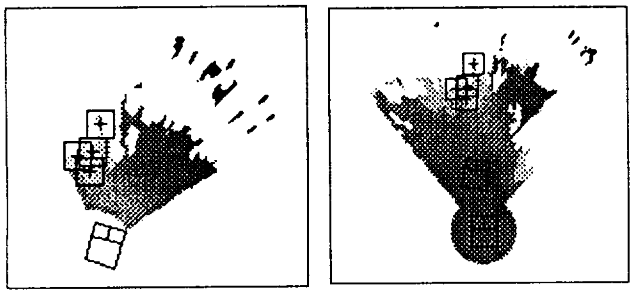


Figure 9: **Position updating : how to merge the new LEM in the current GEM ?**

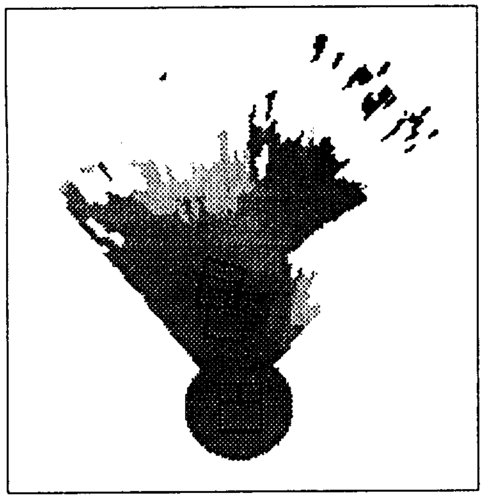


Figure 10: **The new GEM after localization and fusion**

Figure 11 is a perspective view of the reconstructed terrain on which the 3D trajectory of the robot has been planned and executed after 5 incremental steps (the grid discretization of the elevation map is 10 cm). The concatenation of the different 3D trajectories planned by ADAM to reach the goal is surimposed to the terrain model.

7 Conclusion and Future Work

We have presented an integrated multi-level perception system for an autonomous outdoor robot. This system points out several different modeling services, and enhances a lot the robot autonomy and efficiency. An ambitious experimental project, still under way, validates our adaptive approach and benefits to the development of highly demanding robotic applications, in particular planetary exploration.

A lot of difficult tasks have nevertheless still to be achieved, among which we retain the followings :

• Besides the software complete integration of the whole system (and especially of the fine modeling and localization modules), each process performance needs to be improved and better validated. Feedback provided by the real data gathered during the experiments is here an essential information.

• The integration of a stereovision correlation algorithm would enhance the perception capabilities, by providing dense 3D and color data on a particular area. We could then address natural landmark recognition, and estimate the physical nature of the soil during the classification procedure.

• We currently only experimented the 2D navigation mode and the 3D navigation mode apart. Mixing both modes with a reflex one requires the development of "smart" navigation strategies. This topic needs particularly to be better formalized and tested ; the idea of developing exploration strategies in a topological connection graph whose arcs are valued with a certain confidence, while having the possibility of raising up this confidence (by acquiring data), appears to be promising.

• Memory management and consistency management of the models is a bottleneck to the execution of very long missions. The "sliding bitmap" concept we briefly presented has to be implemented and tested.

• Finally, improving the robot speed is fundamental, if not vital. The robot computing capacities should be better exploited, by implementing a kind of "pipeline" architecture.

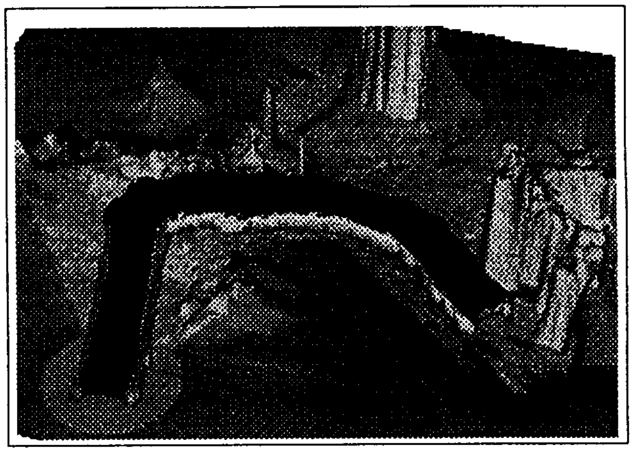


Figure 11: **The GEM after 5 perceptions**

References

[1] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the carnegie-mellon navlab", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 362–373, May 1988.

[2] G. Giralt, R. Chatila, and R. Alami, "Remote Intervention, Robot Autonomy, And Teleprogramming: Generic Concepts And Real-World Application Cases", *in IEEE International Workshop on In-*

telligent Robots and Systems (IROS '93), Yokohama, (Japan), pp. 314–320, July 1993.

[3] G. Giralt, R. Alami, R. Chatila, and P. Freedman, "Remote operated autonomous robots", *International Symposium on Intelligent Robotics, Bangalore (Inde)*, Jan. 1991.

[4] G. Giralt and L. Boissier, "THE FRENCH PLANETARY ROVER VAP: Concept and Current Developments", *in IEEE International Workshop on Intelligent Robots and Systems (IROS '92), Raleigh (North Carolina, USA)*, pp. 1391–1398, July 1992.

[5] F. D. Burke, "Past us studies and developments for planetary rovers", *in Misions, Technologies and Design of Planetary Mobile Vehicles, CNES, Toulouse, France*, Sept. 1992.

[6] R. A. Brooks, "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, Apr. 1986.

[7] R. Chatila, R. Alami, S. Lacroix, J. Perret, and C. Proust, "Planet exploration by robots : from mission planning to autonomous navigation", *in To be published in '93 International Conference on Advanced Robotics*, 1993.

[8] R. Chatila, S. Fleury, M. Herrb, S. Lacroix, and C. Proust, "Autonomous navigation in natural environment", *in Third International Symposium on Experimental Robotics, Kyoto, Japan, Oct. 28-30*, 1993.

[9] S. Lacroix, P. Fillatreau, F. Nashashibi, R. Chatila, and M. Devy, "Perception for Autonomous Navigation in a Natural Environment", *in Workshop on Computer Vision for Space Applications, Antibes, France*, Sept. 1993.

[10] M. Hebert, C. Caillas, E. Krotkov, I.S. Kweon, and T. Kanade, "Terrain Mapping for a Roving Planetary Explorer", *in IEEE International Conference on Robotics and Automation, Scottsdale, (USA)*, pp. 997–1002, 1989.

[11] M. Daily, J. Harris, D. Kreiskey, K. Olion, D. Payton, K. Reseir, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous cross-country navigation with the alv", *in IEEE International Conference on Robotics and Automation, Philadelphia (USA)*, 1988.

[12] E. Schalit, "Arcane : Towards autonomous navigation on rough terrains", *in IEEE International Conference on Robotics and Automation, Nice, (France)*, p. 2568 2575, 1992.

[13] R. Chatila and J.P. Laumond, "Position referencing and consistent world modeling for mobile robots", *in IEEE International Conference on Robotics and Automation, St Louis (USA)*, Apr. 1985.

[14] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach", *in International Journal of Robotics Research*, 1991.

[15] T. Siméon and B. Dacre Wright, "A Practical Motion Planner for All-terrain Mobile Robots", *in IEEE International Workshop on Intelligent Robots and Systems (IROS '93) Japan*, July 1993.

[16] Z. Zhang, "Recalage 3d", Programme vap : Rapport final de phase 5, Institut National de Recherche en Informatique et en Automatique - Sophia Antipolis, 1992.

[17] P. Fillatreau and M. Devy, "Localization of an autonomous mobile robot from 3d depth images using heterogeneous features", *in IEEE International Workshop on Intelligent Robots and Systems (IROS '93), Yokohama, , Japan)*, July 1993.

[18] S. Betge-Brezetz, R. Chatila, and M.Devy, "Natural scene understanding for mobile robot navigation", *in Submitted to IEEE International Conference on Robotics and Automation, San Diego, California*, 1994.

[19] J.-C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[20] F. Nashashibi, M. Devy, and P. Fillatreau, "Indoor Scene Terrain Modeling using Multiple Range Images for Autonomous Mobile Robots", *in IEEE International Conference on Robotics and Automation, Nice, (France)*, pp. 40–46, May 1992.

[21] F. Nashashibi and M. Devy, "3D Incremental Modeling and Robot Localization in a Structured Environment using a Laser Range Finder", *in IEEE Transactions on Robotics and Automation, Atlanta (USA)*, May 1993.

[22] P. Fillatreau, M. Devy, and R. Prajoux, "Modelling of Unstructured Terrain and Feature Extraction using B-spline Surfaces", *in '93 International Conference on Advanced Robotics (ICAR),Tokyo (Japan)*, Nov. 1993.

[23] R. Chatila, R. Alami, B. Degallaix, and H. Laruelle, "Integrated planning and execution control of autonomous robot actions", *in IEEE International Conference on Robotics and Automation, Nice, (France)*, 1992.

[24] R. Alami, R. Chatila, and B. Espiau, "Designing an intelligent control architecture for autonomous robots", *in Third International Symposium on Experimental Robotics, Kyoto, Japan, Oct. 28-30*, 1993.

[25] R. Ferraz De Camargo, R. Chatila, and R. Alami, "A distributed evolvable control architecture for mobile robots", *in '91 International Conference on Advanced Robotics (ICAR),Pisa (Italy)*, pp. 1646–1649, 1991.

[26] T.C. Henderson, C. Hansen, and B. Bhanu, "A framework for distributed sensing and control", *in 9th International Joint Conference on Artificial Intelligence (IJCAI), Los Angeles, California (USA)*, 1985.

[27] M.P.Georgeff and F. F. Ingrand, "Decision-Making in an Embedded Reasoning System", *in 11th International Joint Conference on Artificial Intelligence (IJCAI), Detroit, Michigan (USA)*, 1989.

[28] K. Tarabanis, R.Y. Tsai, and P.K.Allen, "Automated sensor planning for robotic vision tasks", *in IEEE International Conference on Robotics and Automation, Sacramento, (USA)*, pp. 76–82, 1991.

S27-63
207650
P-6

6048

AIAA-94-1202-CP

FUZZY LOGIC BASED ROBOTIC CONTROLLER

N94-30553

F. ATTIA* , M. UPADHYAYA**
* Associate Professor, ** Research Associate

University of Houston, College of Technology
4800 Calhoun Road, Houston, Texas 77204 - 4083

Abstract

Existing Proportional-Integral-Derivative (PID) robotic controllers rely on an inverse kinematic model to convert user-specified cartesian trajectory coordinates to joint variables. These joints experience friction, stiction and gear backlash effects. Due to lack of proper linearization of these effects, modern control theory based on state space methods cannot provide adequate control for robotic systems. In presence of loads, the dynamic behavior of robotic systems is complex and nonlinear, especially where mathematical modeling is evaluated for real-time operations. Fuzzy Logic Control is a fast emerging alternative to conventional control systems in situations where it may not be feasible to formulate an analytical model of the complex system.

Fuzzy logic techniques track a user-defined trajectory without having the host computer to explicitly solve the nonlinear inverse kinematic equations. The goal is to provide a rule-based approach, which is closer to human reasoning. The approach used expresses end-point error, location of manipulator joints, and proximity to obstacles as fuzzy variables. The resulting decisions are based upon linguistic and non-numerical information.

This paper presents a solution to the conventional robot controller which is independent of computationally intensive kinematic equations. Computer simulation results of this approach as obtained from software implementation are also discussed.

Introduction

Fuzzy set theory was developed in 1965 by Zadeh [1], and permits the treatment of vague, uncertain, imprecise, and ill-defined knowledge and concepts in an exact mathematical way. This theory addresses the uncertainty that results from boundary conditions as opposed to Probability theory of mathematics. It allows one to express the operational and control laws of a system, linguistically in words such as "too cold", "cool", "warm", "very hot" etc., which is a generalization of the classical set theory. Fuzzy arithmetic differs from classical Boolean arithmetic as it allows a variable to be partially included in any given set as opposed to being fully included or excluded in Boolean algebra. This is known as **Crisp set theory**. Fuzzy logic is multivalued and varies from maximum to minimum as a function of the input. Fuzzy sets are subjective as compared to standard crisp sets which are objective and are viewed as exceptional cases of fuzzy sets [2].

Fuzzy controllers offer some practical advantages over conventional controllers like increased robustness in spite of high ambient noise levels or sensor failures, an ability to handle nonlinearities without control system degradation, and easy formulation of fuzzy rules. This makes the understanding, modification and maintenance of a fuzzy logic based controller much easier than is possible with conventional controllers. This method can be used when a specific rule base or expert is available who can specify the rules underlying the system behavior and the fuzzy set that represents the characteristics of each variable. The drawbacks of the inverse kinematic equations have posed significant limitations on the robot controller since it is difficult to move the end-effector to a specified position and computing joint variables.

This paper discusses a novel approach in designing a fuzzy logic controller for the robotic arm which replaces the traditional controller and lays the foundation for a new generation of robotic controllers with a simpler architecture.

Conventional Controller Design of Manipulators

The most common controller for robotic manipulators in feedback systems is the Proportional-Integral-Derivative (PID) controller, which is implemented as a secondary controller. This controller corrects errors by means of trajectory tracking [3]. A PID controller performs Proportional amplification

(P),Integration (I), and Differentiation (D) on the error signal fed into the controller as input. In general, the D-part speeds up the response by performing a predictive-type function, I-part influences the steady-state error, and the P-part influences the open-loop steady-state gain. Each part of the controller needs adjustment or tuning experimentally so that desirable responses of the system are obtained. The gain of a PID controller can also be determined by Eigen value assignment.

The PID controller is very simple to implement and each axis can have its own separate PID loop. The main drawback of the PID controller is that the load seen by the motor or actuator of each joint can change rapidly and substantially. This is particularly true for the proximal joints near the base where the moments of inertia and the loading due to gravity can vary by an order of magnitude [4].

Implementation of Fuzzy Logic

A fuzzy logic controller can be considered as a control expert system which simulates human thinking in the interpretation of the real world data. It utilizes fuzzy set labels and performs an appropriate reasoning using Compositional Rule of Inference (CRI) [5]. The CRI represents the core of the deduction mechanism of the controller. It performs the composition of fuzzy sets and matrices of fuzzy rules using the max-min operator. One of the main advantages of using fuzzy approach is that it provides the best technique for knowledge representation that could be possibly devised for encoding knowledge about continuous (analog) variables.

The components of the conventional and fuzzy systems are similar. They differ mainly in fuzzy systems containing the *Fuzzifier* which maps the input physical variables measured by an external sensor to fuzzy set variables [6]. The conditional rules expressed in the form of IF (some event) THEN (perform some action) are contained within the *rule evaluator*. The inverse process of converting the fuzzy outputs of the fuzzy rule evaluator to a physical variable is performed by the *Defuzzifier*. The value produced by the defuzzifier represents the weighted average of all fuzzy rules that were fired within the fuzzy rule evaluator.

The fuzzy system designer's task lies in defining the data points flowing in the system, the basic transformations performed on the data and the data elements output from the system. The first step consists of analyzing the system and understanding the given problem. Next, each control and solution variable in the fuzzy model is decomposed into a set of fuzzy regions. These regions are given unique names, called labels within the domain of the variable. The measured values of input are then converted to corresponding degrees of membership in fuzzy sets. This is done by applying the definition of membership functions for each input variable. Rules that tie the input values to the output model are written as follows : "if < fuzzy proposition A >, then < do fuzzy proposition B >". Generally, the number of rules a system requires is related to the number of control variables. The last step would be to select a method of "defuzzification". There are several ways to convert an output fuzzy set into a crisp solution variable, but the most commonly used one is the centroid technique. Thus the real complexity in developing a fuzzy system is in creating and testing both the degree of membership functions and the rule base, rather than implementing the run-time environment.

Proposed Fuzzy Logic Controller Model

The two basic problems encountered when attempting to apply a fuzzy control in real systems are:

- Choice of primary fuzzy sets to be used together with the rules that constitute the control law or algorithm for a fuzzy control structure.
- Numerical description of the linguistics to implement a fuzzy control algorithm in a computer, which is a nonfuzzy machine.

The typical robot control problem consists of moving the end-effector to a user-specified position (x,y,z) and orientation (roll, pitch, yaw) [7]. To achieve this, the robot joint motors must be driven to specific angular positions. The task of computing these specific joint angles is referred to as the inverse kinematic problem. In general, inverse kinematic equations are highly coupled and involve nonlinear differential equations, whose closed form solutions are often undefined. This poses a computational bottleneck. The block diagram of the proposed Fuzzy Logic Controller is shown in Figure 1.

The Southwestern Research Institute (SWRI) [6] at San Antonio, Texas applied fuzzy logic to control a robot without having to explicitly solve inverse kinematic equations. This controller, mimics intelligent human-like decision-making via a fuzzy rule base, which is essentially a collection of varying degrees of cause-and-effect relationships. The fuzzy rule base is the most critical element within the novel robot controller. The performance of the controller is directly dependent on the quality of fuzzy rules. The approach taken to realize the optimum set of rules which would track enabling control was to linearize the robot model and then apply the principle of superposition to the resulting linearized equations. First, the x and y components of the individual locations of robot joints and the observed tracking error of the robot end-effector need to be represented in fuzzy terms such as: Positive Big (PB), Positive Medium (PM) etc. up to Negative Big (NB). Next, simple fuzzy rules were formulated to evaluate the individual joint axis contributions to reduce the tracking errors of the robot end-effector. For example, if the tracking error in the x direction is PM and the y component of the end-effector is PB, then move the first joint by PM. If robot end point is Negative Medium (NM) and tracking error is Positive Big (PB), change joint angle 1 by NM.

A Simple 2-Degree of Freedom Manipulator

The problem of designing a manipulator controller stems from the basic idea of the simplest known biological controller which is the human arm [8]. When we reach for an object, we determine the approximate error (distance from our hand to the object), and move in a way to reduce the error. We do not precompute the path or the elbow or shoulder angles which is required to grasp the object. Our motions continuously aim at reducing the distance between the hand and the target. In fact, we are successful at reaching and grasping both stationary and moving objects and accomplish these feats without an accurate mathematical model of the kinematics involved. Thus, the fuzzy logic approach allows an initial control system to be derived from fundamental concepts without the need for extended training sets. There are several approaches that achieve this objective. One such approach is discussed in this paper.

The coordinates of the manipulator of the desired point, or target (the end-effector is assumed to be located at the tip of the second link, or at the second joint) are (x_d, y_d), (x_0, y_0) the coordinates of the manipulator of the initial point, e(r) is the error of the manipulator between the initial and the end points, r_d and r_0 are the desired and initial arm lengths (distance from the base joint to the manipulator), angles 180-C,180-D and E are the initial and final angles between the links respectively and the error angle E = C-D, we have:

$$e(r)^2 = r_d^2 - r_0^2$$
$$= 2.L1.L2.(\cos C - \sin C.\ E - \cos C)$$

if angle E is small

$$= 2.L1.L2\ \sin C.\ E$$

where sinE = E for E << 0.

Here, $e(r)^2$ is used as the input signal to the fuzzy set rules.

Actually, $e(r)^2 = [(x_d{}^2 + y_d{}^2) - (x_0{}^2 + y_0{}^2)]$, which reduces the error [9]. After achieving the desired r_d through the change in angle C to angle D, angle A is changed to A' to rotate the robot arm to reach the desired position. The pictorial representation is given in Figure 2.

Here, the rules are arranged as follows:

- For the position of Fig 2(a):
  If(robot arm length needs to be changed by<fuzzy set 1>, and current joint angle is <fuzzy set 2>), then (change second link angle C by E)
- For the position of Fig 2(b):
  If(change in angle C is E, and desired angular change of robot arm length T'-T is <fuzzy set 3>), then (change angle A to A') where <fuzzy set i> (i = 1,2, ...) is of the form "positive big", "small" etc.

The developed fuzzy rule sets reside within the fuzzy controller, which outputs an incremental joint command to the individual joints of the robot based on the configuration and the deviations of the actual end point to the desired end point. The actual Cartesian end point is determined by applying the forward kinematic equations on joint angles [10,11]. The same procedure can be extended to 3 or higher DOF manipulators.

Simulation

The simulation of the proposed algorithm of the above algorithm, was done on a Mach operating system running NExT machine. The trajectory of a robot tracking a user specified straight line and partial configurations are shown in Figs. 3 and 4. The configurations of the robot are all in reasonable good positions, in the sense that those positions keep all joints away from their singular points. It also shows that the robot has passed one of its singular points, which usually causes an overflow in the conventional mathematical algorithm. The error between the actual and the desired trajectory are between specified limits. A computer simulation program is included in the Appendix.

Results of this simulation were graphed, and the performance for the position of the x and y co-ordinates and the error of the arm with respect to time were plotted (Figures 3 & 4). From Figure 3 one can see that the arm was successful in tracking the desired trajectory. Figure 4 shows that the error progressively decreases to zero in the least possible time.

Conclusions

A non algorithmic, model free approach has been developed that relies on a fuzzy rule base to evaluate the required axis motion for the robot. This scheme does not require solution to the inverse kinematic equation to arrive at the joint set points. The fuzzy rule base provides fast execution speed because the fuzzy rules perform simple integer additions and multiplications to evaluate the required axis motion. It can be shown that only a maximum of 15 rules are required to evaluate individual joint axis motion and that a linear relationship exists between the number of rules and the degree of freedom of the robot. The fuzzy logic controller approach is found to be 33% faster than traditional controller methods that require solution to the inverse kinematic equation. However, the fuzzy rule approach cannot achieve the tracking accuracies of the PID controller, since a single fuzzy rule describes a patch in the state space rather than an exact single point.

References

1. Zadeh, L. A. (1965). Fuzzy Sets. Information and control, 8:338-353.
2. T. Terano, K. Asai, M. Sugeno, "Fuzzy systems theory and its applications", Academic Press, Inc., San Diego, California, 1992.
3. J. J. Craig, "Introduction to Robotics, Mechanics and Control", Addison - Wesley Publishing Company, 1986, pp. 242-247.
4. H. Asada, J.J.E. Slotine, "Robot Analysis and Control", J. Wiley, New York, 1986.
5. L. A. Zadeh, "Outline of a New approach to the Analysis of Complex Systems and decision processes", IEEE Transaction Systems, Man and Cybernetics, Vol SMC-3, pp. 28-44, 1973.
6. A. Nedungadi, "A Fuzzy Robot Controller - Hardware Implementation", IEEE International Conference on Fuzzy Systems, 1992.
7. "Applying Fuzzy Logic to Motion Control", Technology Today, Southwest Research Institute (SwRI), San Antonio, Texas, Sept. 1991.
8. N. J. Mandic, E. M. Scharf, and E. H. Mamdani, "Practical application of a heuristic fuzzy rule-based controller to the dynamic control of a robot arm", Proceedings of IEEE Part D, pp. 190-203, July 1985.
9. R. N. Lea, J. Hoblit and Y. Jani, "Fuzzy Logic Based Robotic Arm Control", Proceedings of the IEEE International Conference on Fuzzy Systems,San Francisco, California, March 28-April 1, 1993.
10. L. Sultan and T. Janabi, "The Cognitive bases for the design of a new class of fuzzy logic controllers", Proceedings of NAFIPS-92 International Conference, 1992.
11. L. Sultan and T. Janabi, "Intelligent Fuzzy Controller Development System For Handling Complex Control Problems", Canadian Conference on Electrical and Computer Engineering, 1992.
12. T. Yamkawa, "High speed fuzzy controller hardware system", Second Fuzzy Symposium, Japan, pp. 122-130, 1988.
13. G. V. S. Raju, J. Zhou, "Fuzzy Rule base approach for Robot Motion Control", IEEE International Conference on Fuzzy Systems, 1992.

Appendix

```
/*      C program to compute the trajectory of the 2
         DOF manipulator when the arm is constrained
         to move in a st. line of the form y = -X + 4. */
#define m -1       /* define the slope of the st. line */
#define y_intercept 4
#define A 2        /* define a random x value  */
#define B 2        /* define a y value for the first link */
#define C 4        /* define the initial arm position */
#include <stdio.h>
#include <string.h>
double x_final, y_final;
double x_A, y_B, x_C[500],y_D[500];
double dist1, dist2,armlenght,D,arm1_len;
FILE *fp;

main()
{       double time[500];
        double arm2_len, angle_2;
        int i=0, j;
        double error[500];
        D = (m*C) + y_intercept;
        fp = fopen("datafile","w");
        dist1 = sqrt(pow(A,2) + pow(B,2));
        dist2 = sqrt( pow((A-C),2) + pow((B-D),2) );
        armlenght = sqrt(pow(C,2) + pow(D,2));
        puts("give the co-ordinates of final arm
        position");
        scanf("%d %d", &x_final, &y_final);
        arm1_len = armlenght;
        x_C[i] = C; time[0] = 0; y_D[i] = D;
        error[i] = 0;
        do
        {       x_C[i+1] = x_C[i] + ( C/abs(x_final -
        C));
                y_D[i+1] = m * x_C[i+1] +
        y_intercept;
                arm2_len = sqrt(pow(x_C,2) +
        pow(y_D,2));
                time[i+1] = time[i] + 0.1;
                ++i;
        }while((error[i-1] = abs(x_C[i-1] - x_final)) <
        0.01 && abs(y_D[i-1] - y_final) < 0.01);
        for(j=0; j<=i; ++j)
                {       fprintf(fp, "%d ", time[j]);
                        fprintf(fp, " %d", error[j]);
                        fprintf(fp, " %d", x_C[j]);
                        fprintf(fp, " %d\n", y_D[j]);
                }
}
```

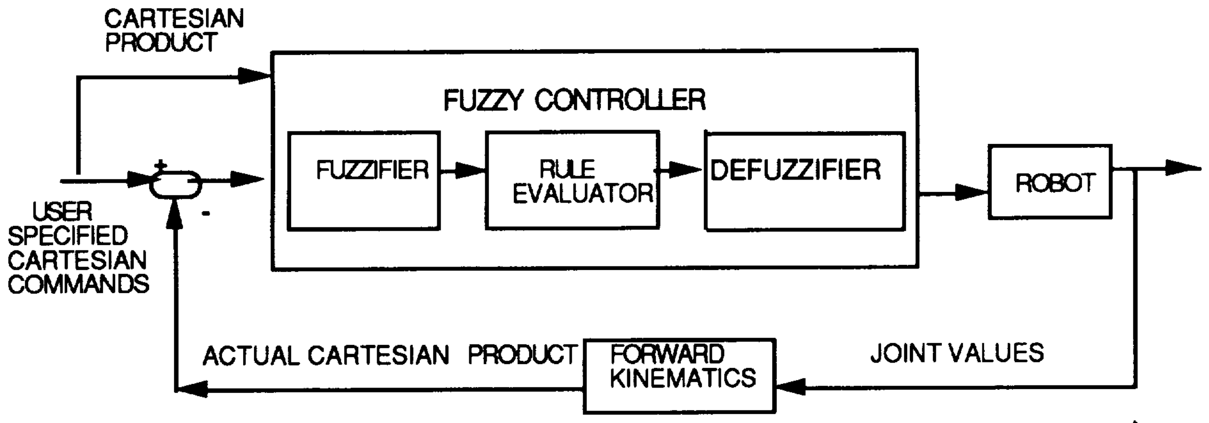


Figure 1. Block Diagram of Proposed Fuzzy Logic Controller

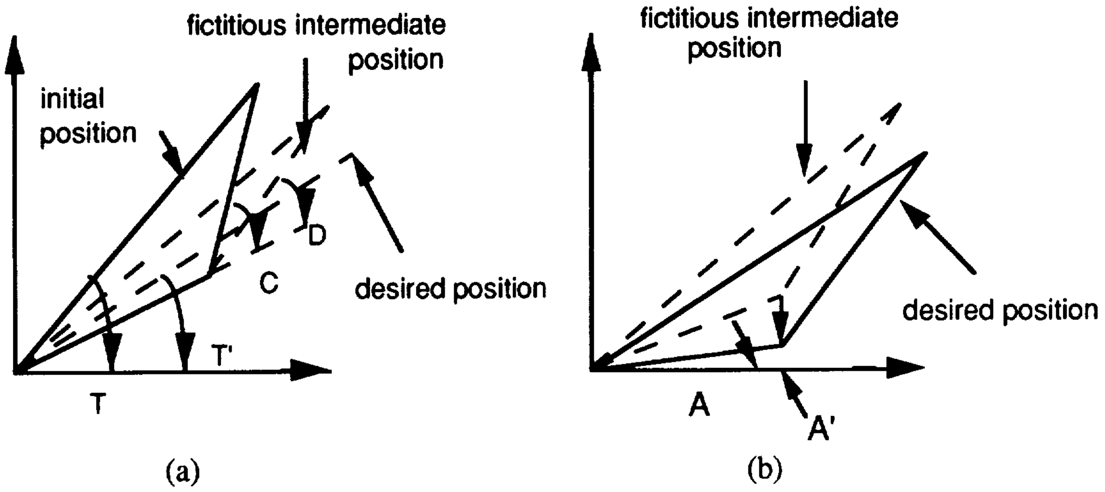

fictitious intermediate position
initial position
D
C
desired position
T'
T
(a)
fictitious intermediate position
desired position
A
A'
(b)

Figure 2. Stretching and Rotating the Robot Arm to Obtain Desired Position

Fig 3: Graph of the x Vs y co-ordinates of the manipulator, as it moves along the preset trajectory path, y = -x + 4 (a straight line).

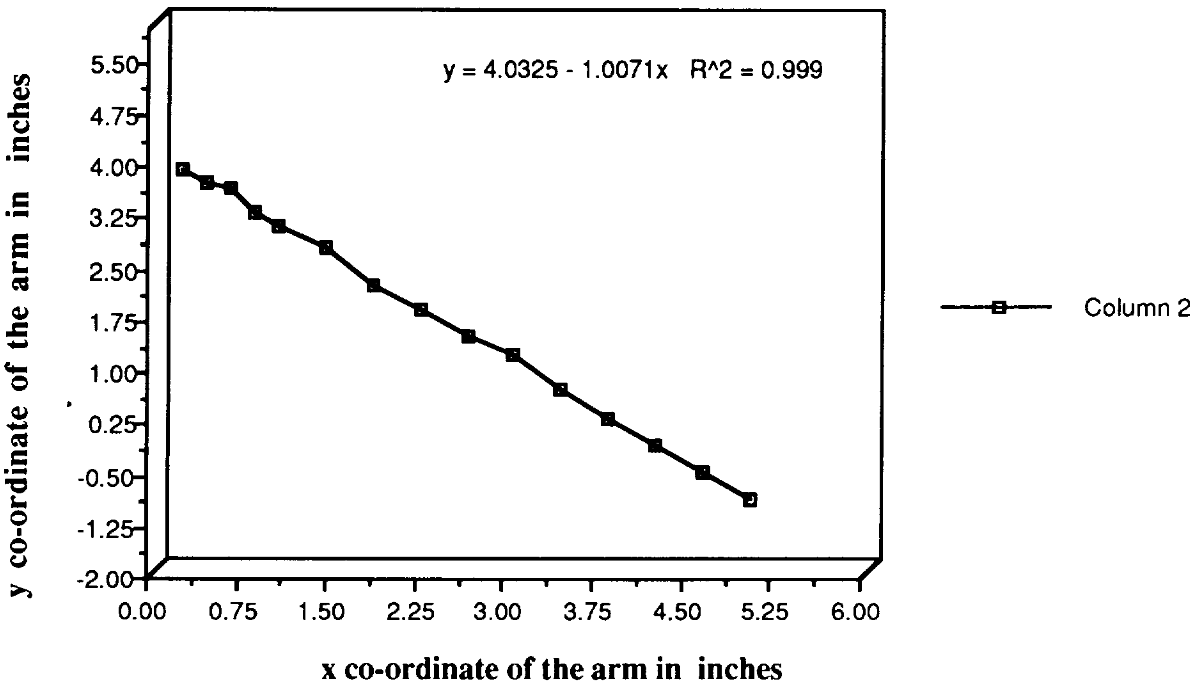
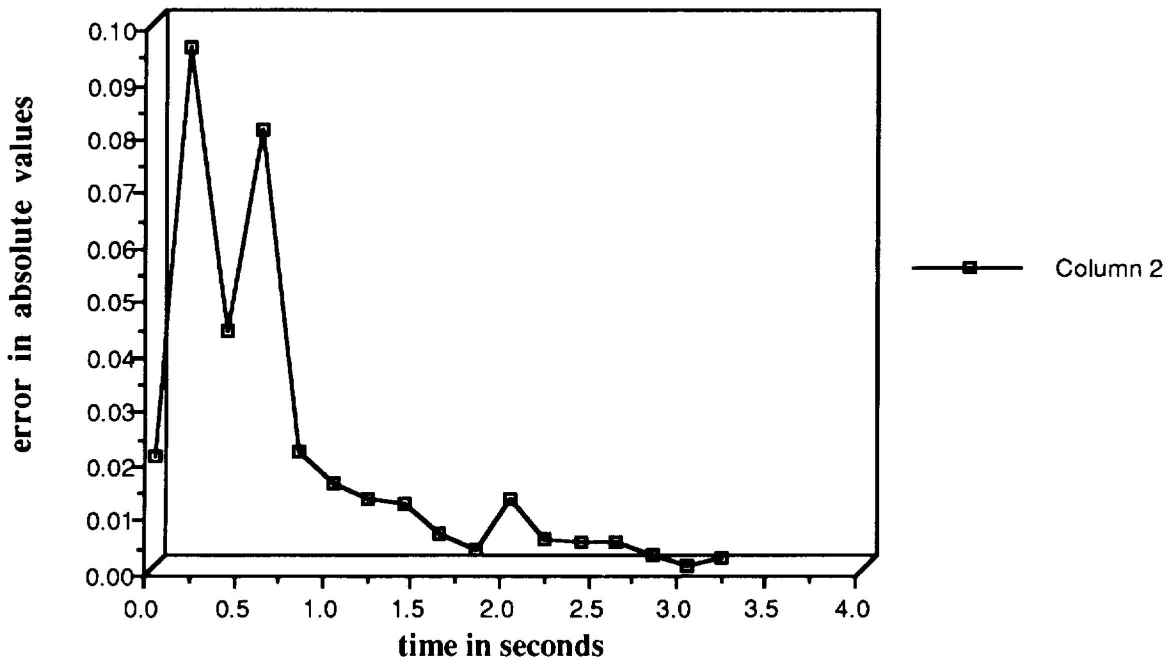


Fig 4: Graph of the error in the position of the manipulator at various time intervals in inches.



S3-63
[illegible]
P- 11

6049

AIAA-94-1203-CP

Vehicle Following Controller Design for Autonomous Intelligent Vehicles

N94- 30554

C. C. Chien, M. C. Lai and R. Mayr
Center for Advanced Transportation Technologies
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562

Abstract. A new vehicle following controller is proposed for autonomous intelligent vehicles. The proposed vehicle following controller not only provides smooth transient maneuver for unavoidable nonzero initial conditions but also guarantees the asymptotic platoon stability without the availability of feedforward information. Furthermore, the achieved asymptotic platoon stability is shown to be robust to sensor delays and an upper bound for the allowable sensor delays is also provided in this paper.

I. Introduction

Designing autonomous intelligent vehicles is important in the research of Advanced Vehicle Control Systems (AVCS) which is a major initiate in Intelligent Vehicle Highway Systems (IVHS). The main advantage of an autonomous intelligent vehicle is that it is considered as a "self-contained" system, i.e., it can operate together with other manually controlled vehicles without further technical assistance from highway infrastructure. Since future Automated Highway Systems (AHS) is planned to evolve from today's highway operation, the deployment of autonomous intelligent vehicles is of particular importance.

An autonomous intelligent vehicle is assumed to be capable of measuring (or estimating) necessary dynamical information from the immediate front vehicle by its on board sensors. The computer in the vehicle will then process these measured data and generate proper throttling and braking actions for controlling the vehicle's movement. These longitudinal maneuvers must be performed as swiftly as possible within the rider's comfort and safety constraints.

Traditionally, vehicle following controllers are designed for single-mass (triple integration) models which do not account for any propulsion system dynamics, see, e.g., [1, 6]. In [8], Shladover included a simple first order engine model in the system dynamics and designed a linear vehicle following controller. It was shown that *asymptotic platoon stability* can be achieved by this linear controller when the drag forces (aerodynamic force and mechanical force) are neglected and the feedforward information is available. Based on the same vehicle model [8] with (nonlinear) drag forces taken into account, a nonlinear vehicle following controller was designed by Sheikholeslam and Desoer [7] using feedback linearization technique. In this case, asymptotic stability can also be achieved if the feedforward information is available. In [3], based on a more complicated vehicle engine model proposed in [5], Hedrick et al. proposed a sliding mode nonlinear controller to achieve vehicle following. The simulation results indicated that the controller has the potential of achieving asymptotic platoon stability if the feedforward information is available. This observation was later verified with proof in [9]. In [4], Ioannou and Chien modified the nonlinear vehicle following controller proposed in [7] and showed that asymptotic platoon stability can be achieved by this modified controller *without* any feedforward information. This result enhances the feasibility of the future deployment of autonomous intelligent vehicles.

In this paper, we propose a new vehicle following controller based on the nonlinear model proposed in [5] and [3]. The proposed vehicle following controller not only provides smooth transient maneuver for unavoidable nonzero initial conditions but also guarantees the asymptotic platoon stability without the availability of feedforward information. Furthermore, we show that the achieved asymptotic platoon stability is robust to sensor delay and an upper bound for the allowable sensor delays is provided.

This paper is organized as follows. In Section 2 and 3 , a vehicle longitudinal model and a safety distance policy are briefly reviewed. In Section 4, we present control methodologies for two classes of nonlinear control systems based on the ideas developed in backstepping control technique. Applying theses methodologies, we design vehicle following throttle and brake controller in Section 5. The issues of designing asymptotic platoon stability and its robustness to sensor delays are discussed in Section 6. In Section 7, we use simulation results to demonstrate the effectiveness of our approach. At last, Section 8 gives a brief conclusion and possible future research directions.

II. Vehicle dynamics model

In this section, we introduce a longitudinal powertrain model for control system design. The derivation of the system dynamic equations is based on the following assumptions [9]:

- Ideal gas law holds in the intake manifold.
- Temperature of the intake manifold does not change.
- There are no time delays in generating the power in the engine.
- The drive axle is sufficiently rigid.
- The torque converter is locked.
- The brakes follow first order dynamics.

The dynamics of the flow of air into and out of the intake manifold is described by

$$\dot{m}_a = \dot{m}_{ai} - \dot{m}_{ao}$$

where m_a is the mass of air in the intake manifold and $\dot{m}_{ai}, \dot{m}_{ao}$ are the mass flow rates through the throttle valve and into the cylinders, respectively.

Empirical equations developed for these flow rates are

$$\begin{aligned} \dot{m}_{ai} &= m_{ax} P_{RI}(m_a) T_C(\alpha) \\ \dot{m}_{ao} &= \dot{m}_{ao}(w_e, m_a) \end{aligned}$$

where m_{ax} is a constant determined by the size of the intake manifold; $T_C(\cdot)$ is the throttle characteristic, a nonlinear function of the throttle angle α; $P_{RI}(\cdot)$ is the pressure influence function describing the choked flow relationship. Notice that $\dot{m}_{a0}$ is generally measured by steady-state engine tests and supplied in tabular form as a function of the mass of air m_a in the intake manifold and the engine speed w_e.

The engine's rotational dynamics is given by

$$I\dot{w}_e = T_{net}(w_e, m_a) - RT_{br} - RrF_{tr} \qquad (1)$$

where I is the rotary inertia of the engine and the wheels referred to the engine side; R is the effective gear ratio from the wheel to the engine; T_{br} is the brake torque; T_{net} is the net-engine torque which is also measured by steady-state engine tests and supplied in tabular form as a function of m_a and the engine speed w_e; r is the effective tire radius; and F_{tr} is the tractive force.

The tractive force can be expressed as

$$F_{tr} = K_r \text{ sat } (i/\bar{i})$$

where K_r is the longitudinal tire stiffness; $\bar{i}$ is a constant determined by the road and tire condition (usually around 0.15 [10]); sat$(\cdot)$ is the standard saturation function; and i is the slip between the wheels and ground given by

$$i = 1 - \frac{v}{Rrw_e}$$

In addition, we adopt a linear brake actuator model

$$\dot{T}_{br} = \frac{T_{bc} - T_{br}}{\tau_b}$$

where τ_b is the actuator time constant, T_{br} is the brake torque applied to the driven wheel and T_{bc} is the commanded brake torque.

Finally, the longitudinal equation for the vehicle velocity is given by

$$M\dot{v} = F_{tr} - cv^2 - \mu Mg \qquad (2)$$

where cv^2 is the aerodynamic drag, μMg is the rolling resistance, and M is the effective mass of the vehicle.

Under the "no-slip" condition [9], i.e.,

$$v = Rrw_e,$$

equations (1) and (2) yield

$$J\dot{w}_e = T_{net}(w_e, m_a) - cR^3 r^3 w_e^2 - RT_{br} - \phi_l$$

where $\phi_l = Rr\mu Mg$; $J = I + Mr^2$ is the effective inertia of the vehicle referred to the engine.

With above discussions, the i^{th} following vehicle has the following longitudinal dynamics,

$$\begin{aligned} \dot{x}_i &= v_i = Rrw_e & (3) \\ \dot{w}_e &= \frac{1}{J}[T_{net}(w_e, m_a) - cR^3 r^3 w_e^2 - RT_{br} - \phi_l] & (4) \\ \dot{m}_a &= -\dot{m}_{ao}(w_e, m_a) + m_{ax} P_{RI}(m_a) T_C(\alpha) & (5) \\ \dot{T}_{br} &= \frac{T_{bc} - T_{br}}{\tau_b} & (6) \end{aligned}$$

where x_i and v_i denote the position and velocity along the longitudinal direction.

III. Safety distance policy

For safe longitudinal operations, a following vehicle is required to keep a safe distance from its preceding vehicle. From the traffic capacity point of view, the desired safe distance should be as small as possible. However, the vehicle's performance capability, rider's comfort constraint and other safety considerations impose minimum bound on this distance. In this paper, we will adopt a desired safety distance policy [4] for the ith following vehicle.

$$S_{d_i} = \lambda_1(v_i^2 - v_{i-1}^2) + \lambda v_i + \lambda_3 \qquad (7)$$

where $\lambda_1, \lambda, \lambda_3$ are positive constants determined by the specified values of human reaction time, vehicle's full acceleration and deceleration, and maximal allowable jerk during deceleration.

While vehicle following is operating near a steady state, the velocity of the control vehicle is approximately equal to the velocity of its preceding vehicle. Therefore, the safety distance policy can be well approximated by the constant time headway policy

$$S_{d_i} = \lambda v_i + \lambda_3. \tag{8}$$

Let x_i (x_{i-1} resp.) and v_i (v_{i-1} resp.) be the position and velocity of the i^{th} ($i - 1^{th}$ resp.) vehicle. As shown in Figure (1), the spacing deviation for the ith vehicle from the desired safety distance is

$$\begin{aligned} \delta_i &:= x_{i-1} - x_i - l_i - S_{d_i} \\ &= x_{i-1} - x_i - l_i - \lambda v_i - \lambda_3 \end{aligned} \tag{9}$$

where l_i is the length of controlled vehicle.

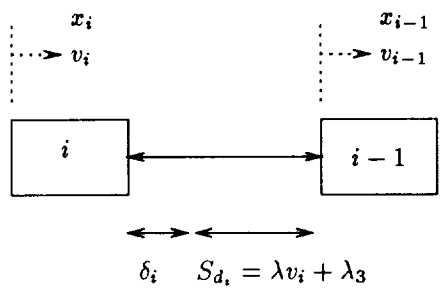


Figure 1:

For a group of vehicles with each vehicle's longitudinal dynamics described by (3) - (6), our control objective is to design a controller for each vehicle such that the following objectives are achieved: the spacing deviation δ_i can be regulated; the asymptotically platoon stability is achieved; and smooth transient response is guaranteed for non-zero initial spacing deviation and velocity deviation.

To this end, it seems that input-output feedback linearization technique may provide a promising approach to deal with this nonlinear control problem based on the structure of the system. However, since the mappings $T_{net}(\cdot,\cdot)$, $\dot{m}_{ao}(\cdot,\cdot)$ and $P_{RI}(\cdot)$ are supplied in tabular forms, their exact partial derivatives are not clearly identified. Consequently, feedback linearization method can not be applied directly.

IV. Nonlinear Control Methodologies

In this section, we will show how the basic ideas used in backstepping control design approach can be applied to controller design for two classes of nonlinear systems. The control methodologies developed will then be used to design vehicle following controllers in Section V..

Nonlinear control systems Class I

Consider the following single-input single-output (SISO) nonlinear control system

$$\begin{aligned} \dot{x} &= f_0(w) \\ \dot{w} &= f_1(w,z) \\ \dot{z} &= f_2(w,z) + f_3(z)g_1(u) \\ y &= h(x,w,x_m,v_m) \end{aligned} \tag{10}$$

where $w, z \in \mathbf{R}$ are state variables; $y \in \mathbf{R}$ is the output; f_i ($i = 0,1,2,3$) and g_1 are smooth nonlinear functions; $x_m, v_m \in \mathbf{R}$ are bounded external signals; and u is the control input.

The control objective is to design the input u so that the output y is regulated, i.e., $\lim_{t\to\infty} y(t) = 0$, while the state variables w, z remain bounded. Our approach for finding an input u to achieved the control objective is based on the application of control Lyapunov function in the backstepping technique developed in [2].

The basic ideas of applying backstepping technique to the control design for system (10) are roughly summarized in the following. First, we neglect the dynamics of state z and treat z as the input, then find a control input $z = z_d$ to achieve output regulation for the following reduced order system:

$$\begin{aligned} \dot{x} &= f_0(w) \\ \dot{w} &= f_1(w,z) \\ y &= h(x,w,x_m,v_m) \end{aligned} \tag{11}$$

Second, construct a state feedback u from the computed z_d such that

$$\lim_{t\to\infty} (z(t) - z_d(t)) = 0$$

Finally, we show the control objective is achieved for the closed loop system.

For our approach, we make the following assumptions.

Assumption 4.1 $f_3(z)$ is nonzero and $\frac{\partial f_1}{\partial z}(w,z)$ is bounded for all $w, z \in \mathbf{R}$.

Assumption 4.2 The system

$$\begin{aligned} \dot{x} &= f_1(x,u) \\ y &= x \end{aligned}$$

is bounded input bounded output (BIBO) stable where $x, y, u \in \mathbf{R}$.

We now elaborate the control design procedure. Take

$$V_w(x,w,x_m,v_m) := \frac{1}{2}\gamma_1 h^2(x,w,x_m,v_m), \quad \gamma_1 > 0$$

as a Lyapunov function and evaluate the derivative of V_w along the trajectory of (11). We get

$$\begin{aligned} &\dot{V}_w(x,w,x_m,v_m) \\ &= \gamma_1 h \frac{dh}{dt} \\ &= \gamma_1 h \left[\frac{\partial h}{\partial x}\dot{x} + \frac{\partial h}{\partial w}\dot{w} + \frac{\partial h}{\partial x_m}\dot{x}_m + \frac{\partial h}{\partial v_m}\dot{v}_m\right] \\ &= \gamma_1 h \left[\frac{\partial h}{\partial x} f_0(w) + \frac{\partial h}{\partial w} f_1(w,z_d) + \frac{\partial h}{\partial x_m}\dot{x}_m + \frac{\partial h}{\partial v_m}\dot{v}_m\right] \end{aligned}$$

If, for $k_1 > 0$, z_d is such that

$$\begin{aligned} f_1(w, z_d) &= (\frac{\partial h}{\partial w})^{-1}[-k_1 h - \frac{\partial h}{\partial x} f_0(w) \\ &\quad - \frac{\partial h}{\partial x_m}\dot{x}_m - \frac{\partial h}{\partial v_m}\dot{v}_m] \end{aligned}$$

then

$$\dot{V}_w(x, w, x_m, v_m) = -\gamma_1 k_1 h^2(x, w, x_m, v_m)$$

For further developments, we will assume

Assumption 4.3 There exists a z_d satisfying (12) for all x, w, x_m, v_m in the domain of interest.

Take

$$\begin{aligned} & V_u(x, w, z, z_d, x_m, v_m) \\ := \ & V_w(x, w, x_m, v_m) + \frac{1}{2}\gamma_2(z - z_d)^2, \quad \gamma_2 > 0. \end{aligned}$$

as a Lyapunov function for (10). The derivative of V_u along the trajectory of (10) is

$$\begin{aligned} & \dot{V}_u(x, w, z, z_d, x_m, v_m) \\ = \ & \gamma_1 h[\frac{\partial h}{\partial x} f_0(w) + \frac{\partial h}{\partial w} f_1(w, z) + \frac{\partial h}{\partial x_m}\dot{x}_m \\ & + \frac{\partial h}{\partial v_m}\dot{v}_m] + \gamma_2(z - z_d)(\dot{z} - \dot{z}_d) \\ = \ & \gamma_1 h[\frac{\partial h}{\partial x} f_0(w) + \frac{\partial h}{\partial w} f_1(w, z_d) + \frac{\partial h}{\partial x_m}\dot{x}_m \\ & + \frac{\partial h}{\partial v_m}\dot{v}_m] + \gamma_1 h \frac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)] \\ & + \gamma_2(z - z_d)[f_2(w, z) + f_3(z) g_1(u) - \dot{z}_d] \\ = \ & -\gamma_1 k_1 h^2 + \gamma_1 h \frac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)] \\ & + \gamma_2(z - z_d)[f_2(w, z) + f_3(z) g_1(u) - \dot{z}_d] \end{aligned}$$

If, for $k_2 > 0$, u is such that

$$\begin{aligned} f_3(z) g_1(u) &= -k_2(z - z_d) - f_2(w, z) + \dot{z}_d \\ &\quad - \frac{1}{\gamma_2(z - z_d)}\gamma_1 h \frac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)] \end{aligned}$$

then

$$\dot{V}_u(x, w, z, z_d, x_m, v_m) = -\gamma_1 k_1 h^2 - \gamma_2 k_2 (z - z_d)^2$$

Theorem 1 *Consider the system (10) with the following proposed nonlinear state feedback controller*

$$\begin{aligned} & u(x, w, z, x_m, v_m) \\ = \ & g_1^{-1}(\tfrac{1}{f_3(z)}\{-k_2(z - z_d) - f_2(w, z) + \dot{z}_d \\ & - \tfrac{\gamma_1}{\gamma_2}\tfrac{1}{z - z_d} h \tfrac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)]\}) \end{aligned} \tag{12}$$

where z_d satisfies (12). Suppose that Assumptions 4.1, 4.2 and 4.3 are satisfied. Then for the closed loop system (10), (12), we have w remains bounded, y converges to zero and z converges to z_d asymptotically.

Proof: Let

$$\bar{z} := z - z_d$$

Then the closed loop syatem (10), (12) yields a subsystem

$$\dot{h} = -k_1 h + \frac{\partial h}{\partial w}[f_1(w, \bar{z} + z_d) - f_1(w, z_d)] \tag{13}$$

$$\dot{\bar{z}} = -k_2 \bar{z} - \frac{\gamma_1}{\gamma_2}\frac{1}{\bar{z}} h \frac{\partial h}{\partial w}[f_1(w, \bar{z} + z_d) - f_1(w, z_d)] \tag{14}$$

By Assumption 4.1, we have

$$\lim_{z \to z_d} \frac{f_1(w, z) - f_1(w, z_d)}{z - z_d} = \frac{\partial f_1}{\partial z}(w, z_d) < \infty$$

This implies that $(h, \bar{z}) = (0, 0)$ is an equilibrium of the system (13), (14).

Take as a Lyapunov function for (13), (14).

$$V(h, \bar{z}) := \frac{1}{2}\gamma_1 h^2 + \frac{1}{2}\gamma_2 \bar{z}^2,$$

which is a positive definite, descrescent, and radially unbounded function. The derivative of V along the trajectory of (13), (14) is

$$\begin{aligned} \dot{V} &= \gamma_1 h \dot{h} + \gamma_2 \bar{z}\dot{\bar{z}} \\ &= -\gamma_1 k_1 h^2 + \gamma_1 h \frac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)] - \gamma_2 k_2 \bar{z}^2 \\ &\quad - \gamma_1 h \frac{\partial h}{\partial w}[f_1(w, z) - f_1(w, z_d)] \\ &= -\gamma_1 k_1 h^2 - \gamma_2 k_2 \bar{z}^2 \\ &< 0 \end{aligned}$$

Therefore, we see

$$h, \bar{z} \in L_2 \cap L_\infty.$$

The boundedness of w can be established by the boundedness of $\bar{z}$ and Assumptions 4.2 and 4.3. Finally, from the well known lyapunov theorem, we conclude that h converges to zero and z converges to z_d asymptotically. □

Nonlinear control systems: Class II We now consider the nonlinear control system

$$\begin{aligned} \dot{x} &= f_0(w) \\ \dot{w} &= f_1(w, z) + f_4(\eta) \\ \dot{z} &= f_2(w, z) \\ \dot{\eta} &= f_3(\eta) + g_1(u) \\ y &= h(x, w, x_m, v_m) \end{aligned} \tag{15}$$

where $x, w, z\eta \in \mathbf{R}$ are state variables; $y \in \mathbf{R}$ is the output; f_i $(i = 0, 1, 2, 3, 4)$ and g_1 are smooth nonlinear functions; $x_m, v_m \in \mathbf{R}$ are bounded external signals; and u is the control input.

The control objective is to design input u so that the output y is regulated while the state variables w, z, η remain bounded. We assume

Assumption 4.4 $\frac{\partial f_4}{\partial \eta}(\eta)$ is bounded.

Assumption 4.5 The system

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2) + f_4(u) \\ \dot{x}_2 &= f_2(x_1, x_2) \\ y &= [x_1 \ x_2]^T \end{aligned}$$

is BIBO stable.

The control design for (15) is similar to the one for system (10). To start with, we neglect the dynamics of state η and treat η as the control input of the system (15), then try to find a control η_d to achieve control objective for the following reduced order system:

$$\dot{x} = f_0(w) \tag{16}$$
$$\dot{w} = f_1(w,z) + f_4(\eta_d) \tag{17}$$
$$\dot{z} = f_2(w,z) \tag{18}$$
$$y = h(x,w,x_m,v_m) \tag{19}$$

Take

$$V_w(x,w,x_m,v_m) = \frac{1}{2}\gamma_3 h^2(x,w,x_m,v_m), \quad \gamma_3 > 0$$

as a Lyapunov function and evaluate the derivative of V_w along the trajectory of (19). We get

$$\begin{aligned}
&\dot{V}_w(x,w,x_m,v_m)\\
&= \gamma_3 h\left[\frac{\partial h}{\partial x}\dot{x} + \frac{\partial h}{\partial w}\dot{w} + \frac{\partial h}{\partial x_m}\dot{x}_m + \frac{\partial h}{\partial v_m}\dot{v}_m\right]\\
&= \gamma_3 h\{\frac{\partial h}{\partial x}f_0(w) + \frac{\partial h}{\partial w}[f_1(w,z) + f_4(\eta_d)]\\
&\quad + \frac{\partial h}{\partial x_m}\dot{x}_m + \frac{\partial h}{\partial v_m}\dot{v}_m\}
\end{aligned}$$

If, for $k_3 > 0$, η_d is such that

$$\begin{aligned}
f_4(\eta_d) &= -f_1(w,z) + \left(\frac{\partial h}{\partial w}\right)^{-1}[-k_3 h - \frac{\partial h}{\partial x}f_0(w)\\
&\quad -\frac{\partial h}{\partial x_m}\dot{x}_m - \frac{\partial h}{\partial v_m}\dot{v}_m]
\end{aligned} \tag{20}$$

then

$$\dot{V}_w(x,w,x_m,v_m) = -\gamma_3 k_3 h^2$$

Similarly, we assume

Assumption 4.6 There exists an η_d satisfying (20) for all x, w, z, x_m, v_m in the domain of interest.

With Assumption 4.6, we take

$$V_u(x,w,\eta,\eta_d,x_m,v_m) = V_w(x,w,x_m,v_m) + \frac{1}{2}\gamma_4(\eta-\eta_d)^2$$

as a Lyapunov function and evaluate its derivative along the system (15). We have

$$\begin{aligned}
&\dot{V}_u(x,w,\eta,\eta_d,x_m,v_m)\\
&= \gamma_3 h\{\frac{\partial h}{\partial x}f_0(w) + \frac{\partial h}{\partial w}[f_1(w,z) + f_4(\eta_d) + f_4(\eta) - f_4(\eta_d)]\\
&\quad + \frac{\partial h}{\partial x_m}\dot{x}_m + \frac{\partial h}{\partial v_m}\dot{v}_m\} + \gamma 4(\eta-\eta_d)(\dot{\eta}-\dot{\eta}_d)\\
&= -\gamma_3 k_3 h^2 + \gamma_3 h\frac{\partial h}{\partial w}[f_4(\eta) - f_4(\eta_d)]\\
&\quad + \gamma 4(\eta-\eta_d)[f_3(\eta) + g_1(u) - \dot{\eta}_d]
\end{aligned}$$

If, for $k_4 > 0$, u is such that

$$\begin{aligned}
g_1(u) &= -k_4(\eta-\eta_d) - f_3(\eta) + \dot{\eta}_d\\
&\quad - \frac{1}{\gamma_4(\eta-\eta_d)}\gamma_3 h\frac{\partial h}{\partial w}[f_4(\eta) - f_4(\eta_d)]
\end{aligned}$$

then

$$\dot{V}_u(x,w,\eta,\eta_d,x_m,v_m) = -\gamma_3 k_3 h^2 - \gamma_4 k_4(\eta-\eta_d)^2$$

Theorem 2 *Consider the system (15) with the following proposed nonlinear state feedback controller*

$$\begin{aligned}
&u(x,w,z,\eta,x_m,v_m)\\
&= g_1^{-1}(-k_4(\eta-\eta_d) - f_3(\eta) + \dot{\eta}_d\\
&\quad - \frac{1}{\gamma_4(\eta-\eta_d)}\gamma_3 h\frac{\partial h}{\partial w}[f_4(\eta) - f_4(\eta_d)])
\end{aligned} \tag{21}$$

where η_d satisfies (20). Suppose that Assumptions 4.4, 4.5 and 4.6 are satisfied. Then for the closed loop system (15), (21), we have w, z remain bounded, y converges to zero and η converges to η_d asymptotically.

Proof: The proof is similar to theorem 1. □

V. Vehicle Following Controller Design

A vehicle following controller is required to maintain a desired spacing between vehicles and to guarantee asymptotic platoon stability. The property that the spacing error for a controlled vehicle can be regulated is referred to *local stability*. A platoon is asymptotically stable if there are no slinky-type effects [7] within a platoon. Researchers have found that local stability in vehicle following is not enough to guarantee asymptotic platoon stability. Moreover, the unavoidable *non-zero initial conditions* occurring during various mode transitions, e.g., switching from manually control to automatic control, can generate transient torque large enough to degrade the driving quality.

In this section, the control methodologies developed in Section IV.are applied to design a vehicle following controller with local stability and asymptotic platoon stability. To deal with the undesirable transient response caused by non-zero initial conditions, we will filter the desired control effort by introducing an imaginary preceding vehicle in the controller design. Stability is guaranteed by the fact that the states of the imaginary preceding vehicle will converge to that of the true preceding vehicle exponentially and the (imaginary) spacing deviation (from the desired spacing between the imaginary vehicle and the controlled vehicle) is regulated. With properly chosen design parameters, the proposed controller achieves asymptotic platoon stability which is robust to sensor delays.

A. Controller Design

The proposed controller is composed of a throttle controller, a brake controller, and a switching logic. The brake controller is to execute the decelerating operation. The throttle controller is to perform the accelerating and decelerating maneuvers while braking is not required for assistance. The switching logic is to properly activate and deactivate the throttle and brake controllers based on the needed control action at the current operating state. To be precise, the controller will continuously compute the required throttle angle required by the control action. If the calculated required throttle angle is greater than the minimum throttle angle, say α_0, the logic determines that the throttle controller alone is capable of handling the desired maneuver, and no brake torque is to be applied. If not, the logic will deactivate the throttle controller, i.e., keep the throttle angle at α_0, and activate the brake controller to generate the proper brake torque.

To smooth the transient response during vehicle maneuvering, we introduce for the i^{th} (following) vehicle an imaginary preceding vehicle with dynamics characterized by the following equations

$$\begin{aligned} \dot{\hat{x}}_{i-1} &= \hat{v}_{i-1} \\ \dot{\hat{v}}_{i-1} &= -\beta_2(\hat{v}_{i-1} - v_{i-1}) - \beta_1(\hat{x}_{i-1} - x_{i-1}) \\ \hat{x}_{i-1}(0) &= x_i(0) + l_i + \lambda v_i(0) + \lambda_3 \\ \hat{v}_{i-1}(0) &= v_i(0) \end{aligned} \tag{22}$$

where $\hat{x}_{i-1}$, $\hat{v}_{i-1}$ can be viewed as the position and velocity of the imaginary proceding vehicle for the i^{th} vehicle; $\beta_1 = \beta_1(\delta_i(0), v_{i-1}(0) - v_i(0))$ and $\beta_2 = \beta_2(\delta_i(0), v_{i-1}(0) - v_i(0))$ are positive functions of $\delta_i(0)$ and $(v_{i-1}(0) - v_i(0))$ to be specified by designers.

Remark 5.1 It is easily verified that if $\dot{v}_{i-1} = 0$, i.e., the (true) preceding vehicle is traveling at constant velocity, it can be easily shown that $(\hat{x}_{i-1} - x_{i-1})(t)$ and $(\hat{v}_{i-1} - v_{i-1})(t)$ converge to 0 exponentially. With suitably chosen parameters β_1 and β_2, we can have proper convergence property of $(\hat{x}_{i-1} - x_{i-1})(t)$ and $(\hat{v}_{i-1} - v_{i-1})(t)$.

Remark 5.2 Negative $\delta_i(0)$ or $v_{i-1}(0) - v_i(0)$ may lead to the situation that the imaginary preceding vehicle is traveling ahead of the true preceding vehicle. For large negative value of $\delta_i(0)$ or $v_{i-1}(0) - v_i(0)$, which is possibly an indication of impending collision, it is necessary to reflect this situation to the controller as soon as possible (which enables the controller of the controlled vehicle to be able to respond it properly for avoiding collision). Therefore, the values of β_1 and β_2 should be chosen in the sense that fast convergence rate is assured.

Define

$$\hat{\delta}_i := \hat{x}_{i-1} - x_i - \lambda v_i - l_i - \lambda_3. \tag{23}$$

Compared (23) with (9), $\hat{\delta}_i$ can be regarded as the deviation of the desired spacing between the imaginary vehicle and the controlled vehicle. Furthermore, we see from (22)

$$\hat{\delta}_i(0) = 0.$$

In order to shape the desired transient response, we adopt the idea of PID control and define a function to be regulated

$$h := c_p\hat{\delta}_i + c_I \int_0^t \hat{\delta}_i \, d\xi + (\hat{v}_{i-1} - v_i) \tag{24}$$

where c_p and c_I are design parameters to be determined.

The design of throttle and brake controllers are discussed separately in the following.

Vehicle following throttle controller

Under the condition that the brake controller is deactivated, the vehicle longitudinal dynamic equations are reduced to

$$\dot{x}_i = v_i = Rrw_e \tag{25}$$

$$\dot{w}_e = \frac{1}{J}[T_{net}(w_e, m_a) - cR^3r^3w_e^2 - \phi_l] \tag{26}$$

$$\dot{m}_a = -\dot{m}_{ao}(w_e, m_a) + m_{ax}P_{RI}(m_a)T_C(\alpha) \tag{27}$$

We see that the system (25) - (27) with output function h given in (24) can be represented by equation (10) with the following variable and function substitutions

$$\begin{aligned} &(x, w, z, u) = (x_i, w_e, m_a, \alpha), \\ &f_0(w) = Rrw, \\ &f_1(w, z) = \frac{1}{J}[T_{net}(w, z) - cR^3r^3w^2 - \phi_l], \\ &f_2(w, z) = -\dot{m}_{ao}(w, z), \\ &f_3(z) = m_{ax}P_{RI}(z), \quad (x_m, v_m) = (\hat{x}_{i-1}, \\ &\hat{v}_{i-1}), \quad g_1(u) = T_c(u), \\ &h = (v_m - Rrw) + c_p(x_m - x - \lambda Rrw) \\ &+c_I \int_0^t (x_m - x - \lambda Rrw)(\xi) d\xi \end{aligned}$$

It is further verified that Assumptions 4.1, 4.2 are satisfied. Besides, the Assumptions 4.3 is also satisfied in the range of operation. By Theorem 1, we propose the following control law

$$\begin{aligned} &\alpha \\ &= T_c^{-1}\Big(\begin{array}{c} \frac{1}{m_{ax}P_{RI}(m_a)}\{-k_2(m_a - m_{a,des}) + \dot{m}_{ao}(w_e, m_a) \\ +\dot{m}_{a,des} + \frac{\gamma_1}{\gamma_2}\frac{(1+\lambda c_p)Rr}{J} \\ \frac{h}{m_a - m_{a,des}}[T_{net}(w_e, m_a) - T_{net}(w_e, m_{a,des}]\} \end{array} \Big) \end{aligned}$$

where $m_{a,des}$ satisfies

$$\begin{aligned} &T_{net}(w_e, m_{a,des}) \\ = \quad &\frac{J}{(1 + \lambda c_p)Rr}[c_p(\hat{v}_{i-1} - v_i) + (c_I + k_1c_p)\hat{\delta}_i \\ &+k_1c_I \int_0^t \hat{\delta}_i \, d\xi \; - \beta_2(\hat{v}_{i-1} - v_i) - \beta_1(\hat{x}_{i-1} - x_i) \\ &+k_1(\hat{v}_{i-1} - v_i)] + \phi_l \end{aligned} \tag{28}$$

From Theorem 1, it is clear to see

Proposition 3 *Consider the system (25)-(27). The controller proposed in (28) - (28) will drive h to zero asymptotically.*

While implementing the control law (28), $\dot{m}_{a,des}$ is to be estimated by finite differencing sampling values of $m_{a,des}$.

We will delay the discussion of the convergence of $\delta_i(t)$ until the brake controller is presented since in both control schemes we can show the same convergence property of $\delta_i(t)$

Vehicle following brake controller

When the brake controller is activated, the throttle angle is kept at the minimum α_0. In this case, the vehicle's dynamics is governed by equations (3) - (6) with α replaced by the constant α_0. Notice that the system (3) - (6) with output function (24) can be represented by (15) with the following variable and function substitutions

$$
\begin{aligned}
&(x,w,z,\eta,u) = (x_i, w_e, m_a, T_{br}, T_{bc}), \quad f_0(w) = Rrw, \\
&f_1(w,z) = \frac{1}{J}[T_{net}(w,z) - cR^3r^3w^2 - \phi_l], \quad f_4(\eta) = -\frac{R}{J}\eta \\
&f_2(w,z) = -\dot{m}_{ao}(w,z), \quad (x_m, v_m) = (\hat{x}_{i-1}, \hat{v}_{i-1}), \\
&f_3(\eta) = -\frac{1}{\tau_b}\eta, \quad g_1(u) = \frac{1}{\tau_b}u, \\
&h = (v_m - Rrw) + c_p(x_m - x - \lambda Rrw) \\
&+c_I \int_0^t (x_m - x - \lambda Rrw)(\xi)d\xi
\end{aligned}
$$

In addition, Assumptions 4.4, 4.5 are satisfied. And the Assumptions 4.6 is also satisfied in the operating range.

To regulate the output function (24), we propose the following brake control law

$$
\begin{aligned}
T_{bc} &= \tau_b[-k_4(T_{br} - T_{br,des}) + \frac{1}{\tau_b}T_{br} + \dot{T}_{br,des} \\
&\quad -\frac{\gamma_3}{\gamma_4}\frac{(1+\lambda c_p)R^2r}{J}h\,]
\end{aligned}
\tag{29}
$$

where

$$
\begin{aligned}
T_{br,des} =& \\
&\tfrac{1}{R}[T_{net}(w_e, m_a) - \phi_l] - \tfrac{J}{R^2r(1+\lambda c_p)}[c_p(\hat{v}_{i-1} - v_i) \\
&+(c_I + k_1c_p)\hat{\delta}_i + k_1c_I \textstyle\int_0^t \hat{\delta}_i\, d\xi - \beta_1(\hat{v}_{i-1} - v_i) \\
&-\beta_2(\hat{x}_{i-1} - x_i) + k_1(\hat{v}_{i-1} - v_i)] + \phi_l
\end{aligned}
$$

By Theorem 3, we see

Proposition 4 *Consider the system (3)-(6) with $\alpha = \alpha_0$ and output function (24). The controller proposed in (29)-(30) will drive h to zero asymptotically.*

Similarly, $\dot{T}_{br,des}$ is to be computed numerically by finite difference sampling values of $T_{br,des}$.

Regulation of δ_i

Recall that our goal is to regulate the spacing deviation δ_i in both throttle and brake control cases. This can be done by properly choosing control parameters c_p, c_I and k_1 as shown in the following.

Let

$$k_3 = k_1.$$

Since the engine/brake dynamics are much faster than the vehicle dynamics (which thus can be neglected in the stage of vehicle performance analysis), the vehicle dynamics of the closed loop system under either throttle control (28), (28) or brake control (29), (30) can be represented by

$$
\begin{aligned}
&\dot{v}_i = Rr\dot{w}_e \\
=\;& \frac{1}{1+\lambda c_p}[-\beta_2(\hat{v}_{i-1} - v_{i-1}) - \beta_1(\hat{x}_{i-1} - x_{i-1}) \\
&+(c_p + k_1)(\hat{v}_{i-1} - v_i) + (c_I + k_1c_p)\hat{\delta}_i \\
&+k_1c_I \int_0^t \hat{\delta}_i\, dz]
\end{aligned}
\tag{30}
$$

From the definition of δ_i (9) and (30), we have

$$
\begin{aligned}
&(1+\lambda c_p)\,\dddot{\hat{\delta}}_i \\
=\;& (1+\lambda c_p)(\ddot{\hat{v}}_{i-1} - \ddot{v}_i - \lambda\,\dddot{v}_i) \\
=\;& (1+\lambda c_p)\ddot{\hat{v}}_{i-1} - [-\beta_2(\dot{\hat{v}}_{i-1} - \dot{v}_{i-1}) - \beta_1(\hat{v}_{i-1} - v_{i-1}) \\
&+(c_p + k_1)(\dot{\hat{v}}_{i-1} - \dot{v}_i) + (k_1c_p + c_I)\dot{\hat{\delta}}_i + k_1c_I\hat{\delta}_i] \\
&-\lambda[-\beta_2(\ddot{\hat{v}}_{i-1} - \ddot{v}_{i-1}) - \beta_1(d\hat{o}tv_{i-1} - \dot{v}_{i-1}) \\
&+(c_p + k_1)(\ddot{\hat{v}}_{i-1} - \ddot{v}_i) + (k_1c_p + c_I)\ddot{\hat{\delta}}_i + k_1c_I\dot{\hat{\delta}}_i]
\end{aligned}
\tag{31}
$$

Therefor, we have the following relationship:

$$
\begin{aligned}
&\hat{\delta}_i(s) \\
=\;& \frac{[(1+\lambda c_p+\lambda\beta_2)s^2 + (\beta_2+\lambda\beta_1)s + \beta_1]\hat{v}_{i-1}(s)}{(1+\lambda c_p)s^3 + (\lambda c_I + \lambda k_1c_p + c_p + k_1)s^2 + (\lambda k_1c_I + c_I + k_1c_p)s + k_1c_I} \\
+\;& \frac{[\lambda\beta_2s^2 + (\beta_2+\lambda\beta_1)s + \beta_1]v_{i-1}(s)}{(1+\lambda c_p)s^3 + (\lambda c_I + \lambda k_1c_p + c_p + k_1)s^2 + (\lambda k_1c_I + c_I + k_1c_p)s + k_1c_I}
\end{aligned}
\tag{32}
$$

Furthermore, from (22), we have stable transfer function

$$\frac{\hat{v}_{i-1}(s)}{v_{i-1}(s)} = \frac{\beta_2s + \beta_1}{s^2 + \beta_2s + \beta_1} \tag{33}$$

From (32) and (33), we conclude that, by properly choosing design parameters c_p, c_I and k_1, we can make $\hat{\delta}_i$ converge to zero if $\dot{v}_{i-1}$ is constant, (i.e., if the preceding vehicle is traveling at constant acceleration) and have satisfactory transient response of $\hat{\delta}_i$.

From the definitions of δ_i (9) and $\hat{\delta}_i$ (23), we see

$$\delta_i = \hat{\delta}_i + x_{i-1} - \hat{x}_{i-1}$$

As pointed out in Remark 4.1, $x_{i-1} - \hat{x}_{i-1}$ will converge to zero exponentially under the condition $\dot{v}_{i-1} = 0$. It follows that δ_i will converge to zero while the preceding vehicle is traveling at constant speed.

VI. Asymptotic Platoon Stability

In this section, we will show that by properly choosing design parameters, the controller proposed in Section 5 can achieve asymptotic platoon stability when it is installed on each vehicle of a group of vehicles (one following another) with safe distance rule (8).

Asymptotic Platoon Stability

Consider a group of vehicles all equipped with the proposed throttle controller (28) and brake controller (29). Since, at steady state of vehicle following,

$$\hat{v}_{i-1} = v_{i-1} \quad \text{and} \quad \hat{\delta}_i = \delta_i,$$

we see from (30)

$$
\begin{aligned}
&\dot{v}_i \\
=\;& \tfrac{1}{1+c_p\lambda}[(c_p + k_1)(v_{i-1} - v_i) + (c_I + k_1c_p)\delta_i \\
&+k_1c_I \textstyle\int_0^t \delta_i(z)dz] \\
=\;& c_p\dot{\delta}_i + (k_1c_p + c_I)\delta_i + k_1c_I \textstyle\int_0^t \delta_i dt + k_1(v_{i-1} - v_i)
\end{aligned}
\tag{34}
$$

Differentiating equation (23) three times and substituting the derivative of v_i by (34), we obtain

$$\begin{aligned}
& \dddot{\delta}_i(t) \\
= \; & \ddot{v}_{i-1} - \ddot{v}_i - \lambda \dddot{v}_i \\
= \; & c_p\ddot{\delta}_{i-1} + (k_1 c_p + c_I)\dot{\delta}_{i-1}(t) + k_1 c_I \delta_{i-1} + k_1(\dot{v}_{i-2} - \dot{v}_{i-1}) \\
& -[c_p\ddot{\delta}_i + (k_1 c_p + c_I)\dot{\delta}_i(t) + k_1 c_I \delta_i + k_1(\dot{v}_{i-1} - \dot{v}_i)] \\
& -\lambda[c_p \dddot{\delta}_i + (k_1 c_p + c_I)\ddot{\delta}_i + k_1 c_I \dot{\delta}_i + k_1(\ddot{v}_{i-1} - \ddot{v}_i)]
\end{aligned}$$

From the above equation, we obtain the transfer function from δ_i to δ_{i-1}

$$\frac{\delta_i(s)}{\delta_{i-1}(s)} := G_1(s) =$$
$$\frac{(k_1 + c_p)s^2 + (k_1 c_p + c_I)s + k_1 c_I}{(1 + \lambda c_p)s^3 + (\lambda c_I + \lambda k_1 c_p + c_p + k_1)s^2 + (\lambda k_1 c_I + c_I + k_1 c_p)s + k_1 c_I} \tag{35}$$

To avoid slinky-type effects, the disturbances caused by the lead vehicle in all frequencies should be attenuated along the following vehicles to insure that they do not become unreasonably large by the end. A sufficient condition for this to happen is for all i

$$\left|\frac{\delta_i(jw)}{\delta_{i-1}(jw)}\right| = |G_1(jw)| < 1, \quad \text{for all } w > 0 \tag{36}$$

With $G_1(s)$ given in (35), the inequality in (36) yields

$$\left|\frac{[k_1 c_I - (k_1 + c_p)w^2]^2 + w^2(k_1 c_p + c_I)^2}{[k_1 c_I - (\lambda c_I + \lambda k_1 c_p + c_p + k_1)w^2]^2 + w^2[\lambda k_1 c_I + c_I + k_1 c_p) - (1 + \lambda c_p)w^2]^2}\right|$$
$$< 1 \quad \text{for all } w > 0$$

Simplifying the above inequality, we get

$$\begin{aligned}
& (1 + \lambda c_p)^2 w^4 + [\lambda^2 c_I^2 + \lambda^2 k_1^2 c_p^2 \\
& + 2\lambda k_1 c_p^2 - 2(c_I + k_1 c_p)]w^2 + \lambda^2 k_1^2 c_I^2 > 0 \\
& \text{for all } w > 0
\end{aligned} \tag{37}$$

A sufficient condition such that (37) holds is

$$\lambda^2 c_I^2 + \lambda^2 k_1^2 c_p^2 + 2\lambda k_1 c_p^2 - 2(c_I + k_1 c_p) > 0$$

or equivalently

$$\frac{(c_I - \frac{1}{\lambda^2})^2}{A^2} + \frac{(c_p - \frac{k_1}{\lambda^2 k_1^2 + 2\lambda k_1})^2}{B^2} > 1 \tag{38}$$

where

$$\begin{aligned}
A^2 &= \frac{2\lambda k_1 + 2}{\lambda^4(\lambda k_1 + 2)} \\
B^2 &= \frac{2\lambda k_1 + 2}{\lambda^3 k_1(\lambda k_1 + 2)}
\end{aligned}$$

Given $k_1 > 0$ and $\lambda > 0$, the suitable values of parameters c_I and c_p satisfying inequality (38) reside outside shaded ellipse as shown in Figure 2. Consequently, if we choose c_I, k_1, c_p outside the shaded ellipse as shown in Figure 2, asymptotic platoon stability can be assured.

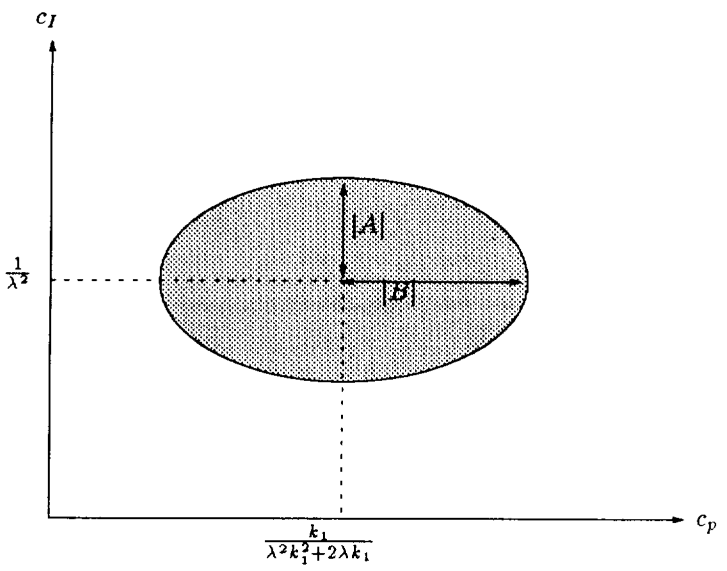


Figure 2: Parameter region for avoiding slinky effects

Remark 6.1 When constant spacing safety policy ($\lambda = 0$) is adopted, inequality (37) for avoiding slinky-type effects reduces to

$$w^2 - 2(c_I + k_1 c_p) > 0$$

Since $c_I + k_1 c_p > 0$ (to insure all the poles of $G_1(s)$ are in the open left half complex plane), the above inequality can not be satisfied when $w^2 < 2(c_I + k_1 c_p)$. In other words, asymptotic stability can not be assured for low frequency disturbances under constant spacing safety distance policy.

Asymptotic platoon stability under sensor delays In this subsection, the relationships between the sensor delays, the gains of the proposed controller, and the asymptotic platoon stability will be investigated. The results obtained in this subsection can be used to quantify the performance requirements for the sensors for a specific designed controller.

Let τ be the time delay caused by the velocity sensor and the position sensor, such that the velocity and position terms in (28) and (30) are functions for $t - \tau$ instead of t. Then the vehicle dynamics of the closed loop system can be represented by

$$\begin{aligned}
\dot{v}_i(t) \; = \; & \frac{1}{1 + c_p\lambda}[(c_p + k_1)(v_{i-1} - v_i)(t - \tau) + (c_I + k_1 c_p)\delta_i(t - \tau) \\
& + k_1 c_I \int_0^t \delta_i(\xi - \tau)d\xi]
\end{aligned} \tag{39}$$

Differentiating both sides of(9) three times, we get

$$\begin{aligned}
& \dddot{\delta}_i(t) \\
= \;& \ddot{v}_{i-1}(t) - \ddot{v}_i(t) - \lambda \dddot{v}_i(t) \\
= \;& c_p \dddot{\delta}_{i-1}(t-\tau) + (k_1 c_p + c_I)\dot{\delta}_{i-1}(t-\tau) \\
& + k_1 c_I \delta_{i-1}(t-\tau) + k_1(\dot{v}_{i-2}(t-\tau) - \dot{v}_{i-1}(t-\tau)) \\
& - [c_p \ddot{\delta}_i(t-\tau) + (k_1 c_p + c_I)\dot{\delta}_i(t-\tau) + k_1 c_I \delta_i(t-\tau) \\
& + k_1(\dot{v}_{i-1}(t-\tau) - \dot{v}_i(t-\tau))] - \lambda[c_p \dddot{\delta}_i(t-\tau) \\
& + (k_1 c_p + c_I)\ddot{\delta}_i(t-\tau) + k_1 c_I \dot{\delta}_i(t-\tau) \\
& + k_1(\ddot{v}_{i-1}(t-\tau) - \ddot{v}_i(t-\tau))]
\end{aligned} \tag{40}$$

Substituting (39) into (40) and taking Laplace transforms, we can derive the transfer function from δ_{i-1} to δ_i

$$\frac{\delta_i(s)}{\delta_{i-1}(s)} := G_2(s) = \frac{(k_1 + c_p)s^2 + (k_1 c_p + c_I)s + k_1 c_I}{(e^{s\tau} + \lambda c_p)s^3 + (\lambda c_I + \lambda k_1 c_p + c_p + k_1)s^2 + (\lambda k_1 c_I + c_I + k_1 c_p)s + k_1 c_I} \tag{41}$$

A sufficient condition for asymptotic stability is, for all i

$$|G_2(jw)|^2 < 1, \quad \text{for all} w > 0$$

Substituting (41) into the above inequality, we obtain

$$\left| \frac{(d - ew^2)^2 + f^2 w^2}{[d - bw^2 + (\sin w\tau)w^3]^2 + w^2[c - (a + \cos(w\tau)w^2]^2} \right| < 1, \quad \text{for all} w > 0 \tag{42}$$

where

$$\left.\begin{aligned}
a &= \lambda c_p, \quad b = \lambda c_I + \lambda k_1 c_p + c_p + k_1 \\
c &= \lambda k_1 c_I + c_I + k_1 c_p, \quad d = k_1 c_I \\
e &= k_1 + c_p, \quad f = k_1 c_p + c_I
\end{aligned}\right) \tag{43}$$

With equations in (43), condition for asymptotic platoon stability (42) is equivalent to

$$\begin{aligned}
& [a^2 + 2a\cos(w\tau) + 1]w^4 - 2b\sin(w\tau)w^3 \\
& + [b^2 - 2ac - 2c\cos(w\tau) - e^2]w^2 + 2d\sin(w\tau)w \\
& + (c^2 - f^2 - 2bd + 2de) > 0, \quad w > 0
\end{aligned} \tag{44}$$

Proposition 5 *Consider the vehicle longitudinal system (3) - (6) with control law (28), (29). The asymptotic platoon stability is guaranteed if*

$$\tau < \min\left\{ \frac{(\lambda c_p - 1)^2}{2(\lambda c_I + \lambda k_1 c_p + c_p + k_1)}, \frac{\lambda^2 c_I^2 + \lambda^2 k_1^2 c_p^2 + 2\lambda k_1 c_p^2 - 2(c_I + k_1 c_p)}{2k_1 c_I} \right\} \tag{45}$$

Remark 6.2 In Subsection 4.2, we have chosen $\lambda^2 c_I^2 + \lambda^2 k_1^2 c_p^2 + 2\lambda k_1 c_p^2 - 2(c_I + k_1 c_p)$ to be positive to insure asymptotic platoon stability. Furthermore, $\lambda c_I + \lambda k_1 c_p + c_p + k_1$ and $k_1 c_I$ are also chosen to be positive to guarantee local stability (regulation of δ_i). Therefore, the right hand side of inequality (45) is positive.

Proof: Since the inequality in (44) can be rewritten as

$$\begin{aligned}
& \{[a^2 + 2a\cos(w\tau) + 1]w - 2b\sin(w\tau)\}w^3 \\
& + \{[b^2 - 2ac - 2c\cos(w\tau) - e^2]w + 2d\sin(w\tau)\}w \\
& + (c^2 - f^2 - 2bd + 2de) > 0,
\end{aligned}$$

asymptotic platoon stability is guaranteed if

$$[a^2 + 2a\cos(w\tau) + 1]w - 2b\sin(w\tau) > 0, \tag{46}$$

$$[b^2 - 2ac - 2c\cos(w\tau) - e^2]w + 2d\sin(w\tau) > 0, \text{ and} \tag{47}$$

$$c^2 - f^2 - 2bd + 2de > 0. \tag{48}$$

It is easily verified that

$$\frac{(a-1)^2}{2b} = \frac{(\lambda c_p - 1)^2}{2(\lambda c_I + \lambda k_1 c_p + c_p + k_1)}$$

and

$$\frac{b^2 - 2ac - 2c - e^2}{2d} = \frac{\lambda^2 c_I^2 + \lambda^2 k_1^2 c_p^2 + 2\lambda k_1 c_p^2 - 2(c_I + k_1 c_p)}{2k_1 c_I}$$

such that condition (45) is equivalent to

$$< \min\left\{ \frac{(a-1)^2}{2b}, \frac{b^2 - 2ac - 2c - e^2}{2d} \right\}. \tag{49}$$

Since $a > 0$ and $b > 0$, we see from (49)

$$a^2 + 2a\cos(w\tau) + 1 \geq a^2 - 2a + 1 > 2b\tau \geq 2b\frac{\sin(\tau w)}{w},$$

$$\begin{aligned}
b^2 - 2ac - 2c\cos(w\tau) - e^2 &\geq b^2 - 2ac - 2c - e^2 > 2d\tau \\
&\geq -2d\frac{\sin(\tau w)}{w}
\end{aligned}$$

which guarantee the inequalities (46) and (47). Moreover, from (43), we see

$$c^2 - f^2 - 2bd + 2de = \lambda^2 k_1^2 c_I^2 > 0$$

which assures the inequality (48). □

VII. Simulation Results

We consider vehicles following each other in a single lane with no passing. Each vehicle is assumed to be equipped with the proposed controller. The length of vehicles is assumed to be 4 meters. The following controller gains were selected for the simulations :

$$c_p = 2, \; c_I = 0.5, \; c_v = 2,$$
$$k_1 = 5, \; k_2 = 40, \; k_3 = 5, \; k_4 = 1 \;, \lambda = 1, \; \lambda_3 = 2.$$

Case 1: Vehicle following with zero initial conditions: Six vehicles are assumed to follow each other and form a platoon in a single lane. The leading vehicle is assumed to accelerate from 9 m/sec to 15 m/sec, then to 21 m/sec, and then to 27 m/sec. After achieves 27 m/sec, it then decelerates to 21.5 m/sec and then to 17 m/sec. Zero-initial conditions are assumed. The simulation results are shown in Figure (3). Good velocity tracking, small transient spacing error and zero steady state spacing error are achieved for each vehicle. Moreover, no slinky-type effects exist. In other words, asymptotic platoon stability is achieved.

Case 2 : Exit from the automatic lane: The following situation is considered : at time $t = 0$ sec, the leading vehicle changes lanes and the new vehicle target is 3.2 m/s faster and meters farther ahead than the previous one. In this situation, a suddenly change of the relative velocity and relative distance appears which is then confirmed by the on-board computer and the automatic control equipment is reset. Thus, non-zero initial conditions appear. The velocity, acceleration, and spacing deviation profiles shown in Figure (4) are quite smooth during the transient stage.

VIII. Conclusion

In this paper, we have studied the vehicle following control problem for the autonomous intelligent vehicles under the constant time headway safety distance rule. Instead of using simplified linear vehicle following models frequently used in vehicle longitudinal control, we consider a nonlinear model that contains important attributes of engines dynamics. Using a newly developed nonlinear control technique, we are able to design throttle and brake controllers for the longitudinal control purpose with smooth maneuvers. One of features of this design is that the asymptotic platoon stability can be achieved with properly chosen design parameters. We further show that this nice property is theoretically robust to a certain degree of sensor delays. The computer simulation results demonstrate the effectiveness of our control approach and enhance the feasibility of practical AICC technology deployment.

Acknowlegement The authors would like to thank Professor Hedrick and Mr. Swaroop for providing the data of the vehicle engine system and several useful comments.

References

[1] H. Y. Chiu, G. B. Stupp, and S. J. Brown. Vehicle-follower controls with variable-gains for short-headway automated guideway transit systems. *ASME Journal of Dynamic System, Measurement and Control*, 99:183–189, 1977.

[2] R. A. Freeman and P. V. Kokotovic. Backstepping design of robust controllers for a class of nonlinear systems. *preprint*, 1991.

[3] J. K. Hedrick, D. McMahon, V. Narendran, and D. Swaroop. Longitudinal vehicle controller design for ivhs systems. *American Control Conference*, pages 3107–3112, 1991.

[4] P. Ioannou and C. C. Chien. Autonomous intelligent cruise control. *to appear in IEEE Transactions on Vehicular Technology*, 1993.

[5] J. J. Moskwa and J. K. Hedrick. Modeling and validation of automotive engines for control algorithm development. *ASME Journal of Dynamic System, Measurement and Control*, pages 278–285, 1992.

[6] A. J. Pue. A state-constraint approach to vehicle-follower control for short headway automated transit vehicles. *ASME Journal of Dynamic System, Measurement and Control*, 99:183–189, 1977.

[7] Shahab Sheikholeslam and Charles A. Desoer. A system level study of the longitudinal control of a platoon of vehicles. *ASME Journal of Dynamic System, Measurement and Control*, 114:286–292, 1992.

[8] S. Shladover. Operation of automated guideway transit vehicles in dynamically reconfigured trains and platoons. *UMTA-MA-06-0085-79*, 1979.

[9] D. Swaroop, C. C. Chien, P. Ioannou, and J. K. Hedrick. A comparision of spacing and headway control laws for automatically controlled vehicle. *submitted to Journal of Vehicle System Dynamics*, 1993.

[10] J. Y. Wong. Theory of ground vehicles. *John Wiley & Sons*, NY, 1978.

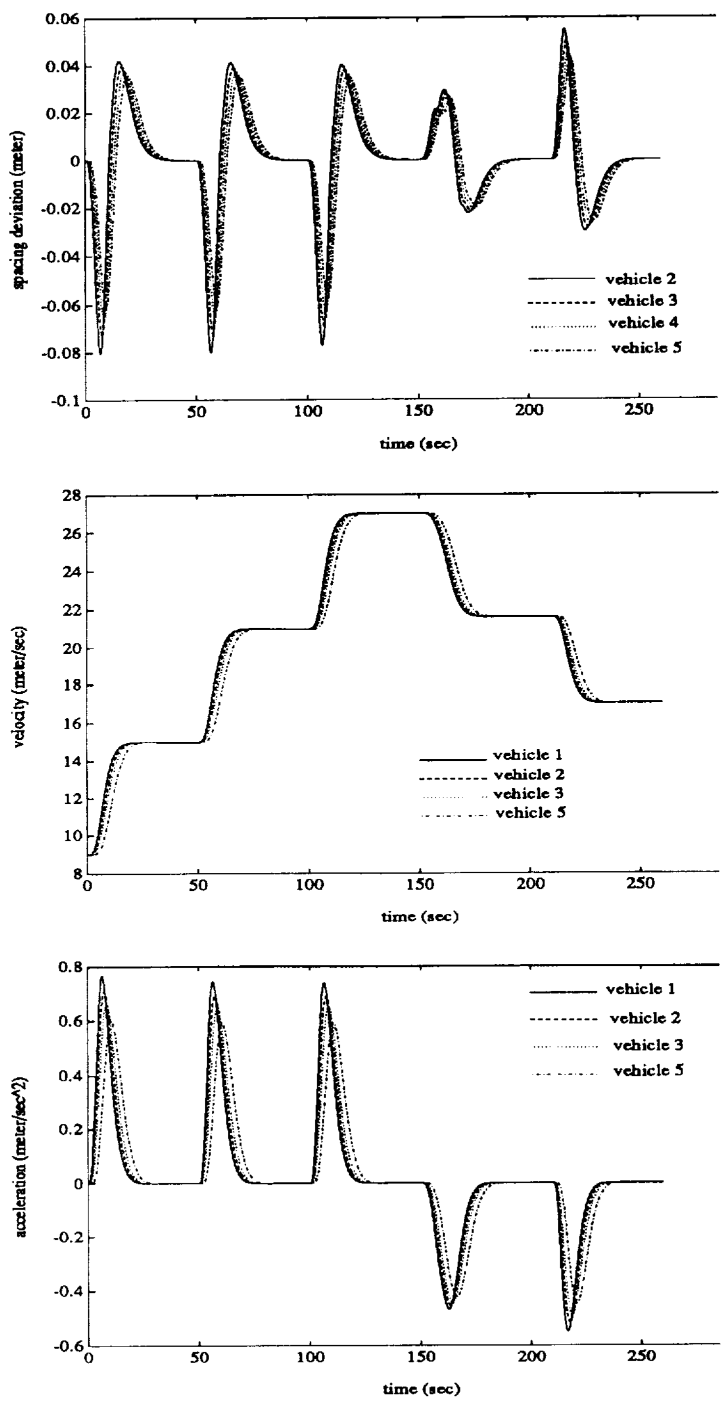


Figure 3: Case 1: Spacing deviation, velocity and acceleration profiles for a vehicle following maneuver with zero initial conditions

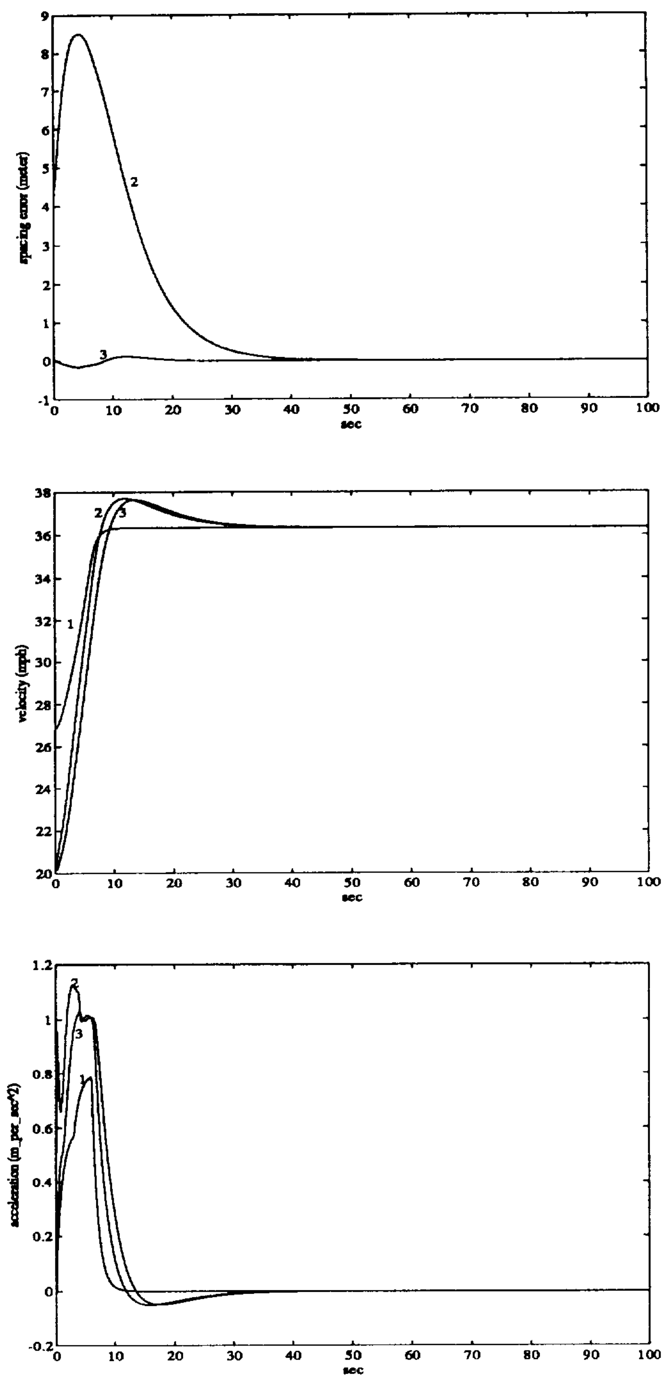


Figure 4: Case 2: Auto-exit situation

6050

AIAA-94-1204-CP

N94- 30555

The Real-World Navigator

Marko Balabanović [*] Craig Becker [†] Sarah K. Morse [‡] Illah R. Nourbakhsh [‡]

Department of Computer Science
Stanford University
Stanford, CA 94305
lightning@cs.stanford.edu

[illegible]

Abstract

The success of every mobile robot application hinges on the ability to navigate robustly in the real world. The problem of robust navigation is separable from the challenges faced by any particular robot application. We offer the *Real-World Navigator* as a solution architecture that includes a path planner, a map-based localizer, and a motion control loop that combines reactive avoidance modules with deliberate goal-based motion. Our architecture achieves a high degree of reliability by maintaining and reasoning about an explicit description of positional uncertainty. We provide two implementations of real-world robot systems that incorporate the Real-World Navigator. The Vagabond Project culminated in a robot that successfully navigated a portion of the Stanford University campus. The SCIMMER project developed successful entries for the AAAI 1993 Robotics Competition, placing first in one of the two contests entered.

1 Introduction

Current research on autonomous mobile robots has highlighted the difficulty of building robust, general-purpose navigation software. Problems with current systems include specificity for a particular environment, inability to deal with dynamic, real-world situations, and short life-spans, often due to the problems of cumulative sensory and control error.

We are studying the problem of robust navigation in the context of problems which can be decomposed as shown in Figure 1. In this decomposition, there is a task level, which provides the navigator level with a series of goals, and there is a physical robot capable of sensing and moving in the world. The navigator level directs the physical robot to achieve the goals of the task level while guaranteeing robust and reliable operation.

In this paper we describe a navigator level architecture called the *Real-World Navigator* that achieves robust robot control in a variety of environments. Given no domain-specific knowledge beyond a floor map, this Navigator should be able to move about an arbitrary office environment while preserving its sense of position.

The sharp decomposition of Figure 1 allows us to use the Real-World Navigator with different physical robots and in different task domains. We will describe two successful implementations, involving different robots in several task domains and both indoor

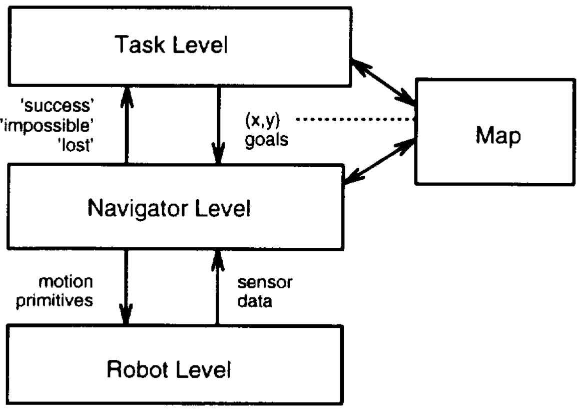


Figure 1: A three level decomposition of a mobile robot system

[*]Funding provided by ARPA order 8607, grant NASA NAG 2-581

[†]Supported by an NSF Graduate Fellowship, research funded by ARPA grant N00014-92-J-1809

[‡]Funding provided by the Office of Naval Research under contract number N00014-90-J-1533

and outdoor environments.

1.1 Assumptions

In the descriptions of the architecture in the remainder of this paper, we make the following assumptions:

1. The system as a whole can be represented according to the interaction paradigm illustrated by Figure 1.
2. The goal coordinates that are passed down from the task level refer to locations in a shared map with bounded error.
3. The Navigator must have bounds on the error of the sensory and motion primitives through which it controls the robot level.
4. The control and sensory latencies of the robot level are appropriate to the dynamics of the environment; it is physically capable of responding to events and maintaining its safety in real time.
5. Any objects that are invisible to the robot's sensors must be present on the map. For instance, our robots have no way of detecting potentially deadly stairwells, so to ensure their (and our!) safety these areas must be marked on the map.

We make no further assumptions concerning the task or robot levels. For instance, it is possible for the task level to be a human operator.

1.2 Goals

The navigator level is an interface between the high-level goals of the robot system and the uncertainties and errors of the real world. As such, it must achieve the high-level position requests whenever they are reachable and, in the case of unreachable goals, it must signal failure. In addition, we expect the Navigator to react gracefully to a dynamic environment by avoiding both mapped obstacles and unmapped, visible obstacles in a smooth and efficient manner.

1.3 Overview

In the next section we present the general architecture of the Real-World Navigator without commitment to any specific task or physical robot. We then describe two implementations of the architecture with which we have solved various navigation tasks on different robot platforms. Next we discuss the limitations of the current architecture as well as extensions that may increase its robustness and applicability. Finally, we summarize work related to ours and present our conclusions.

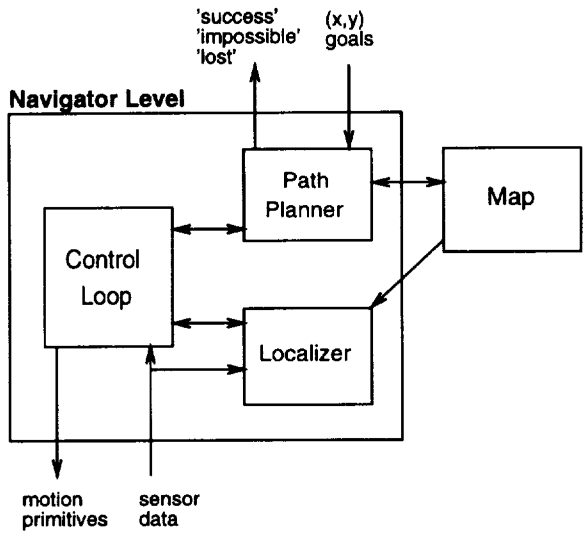


Figure 2: The Navigator consists of three subsystems: a path planner, a control loop, and a localizer. It also references an external map resource.

2 The Real-World Navigator Architecture

We consider the navigator level to be a collection of subsystems which communicate in a well-defined way. Figure 2 depicts the interaction of the subsystems that comprise the Navigator. Arrows in the figure represent data flow between the subsystems as well as between subsystems and the task and robot levels.

Briefly, the execution of a navigation task is as follows: the path planner receives goal coordinates from the task level. It then generates an appropriate plan using information from the map and invokes the control loop to execute each segment of the plan in turn. The control loop interacts with the physical robot and, if necessary, the localizer in order to reliably navigate each path segment. The localizer refers to both the raw sensor data of the robot and the geometric map.

We now discuss each of these subsystems in more detail.

2.1 Map

The map is a shared resource that is externally specified but referenced and manipulated by both the task and navigator levels. It maintains two different representations of the environment: one geometric, and the other based on the concept of highways.

The geometric representation is simply any description of the obstacles and free-space using an appropriate and agreed-upon coordinate system. For example, a reasonable geometric map for a robot that moves in a plane would be a polygonal representation of the projection of obstacles onto that plane. Note that this should be a map of physical space rather than configuration space because the localizer will compare the geometric map to sensor data.

In addition, the Navigator makes use of a highway-based representation of the map. The idea behind highways is to constrain the possible motions of the robot, both to simplify planning and to reduce the number of features that the robot must reliably sense.

Definition 1 (Highway Constraint) *Highways are possibly overlapping regions which decompose a subset of the free space of the robot's environment. The robot must always move within highways, and therefore can move between highways only through regions where they overlap.*

This constraint is related to highways in the real world. For example, planning a trip from San Francisco to Los Angeles would be much harder if we considered every possible back road instead of staying on the interstates. Using the interstates also means that we need only recognize off-ramps to move from one highway to another, rather than all the myriad types of intersection we might otherwise encounter.

Note that the highway map can either be provided by a human or automatically generated from the geometric map. Both methods have advantages. A human might want to design the highways to limit the robot's motion to certain parts of the free space (for example, to avoid a particularly busy hallway) or to hand-optimize certain motions. On the other hand, automatic generation of highways could save tedious work. There are several classical algorithms from motion planning that may be useful in automatic highway generation; examples are cell decomposition and visibility graph construction [Latombe, 1991].

2.2 Path Planner

Given the map and a goal position from the task planner, the function of the path planner is to compute a list of interim points through which the robot can move to achieve the goal. These interim points are passed in turn to the control loop, which guides the robot to each sub-goal. We assume that the path planner uses its knowledge of the geometric map to ensure that the points on this list can safely be connected by straight-line paths. Of course this assumption may be false in the face of unknown obstacles, but handling that contingency is the responsibility of the control loop which we describe below. We also assume that the path planner respects the constraints that the highway map imposes. Specifically, each of the interim straight-line sub-paths must lie completely within a highway.

Note that the choice of highway representations will influence the complexity of the path planner. For instance, suppose that we define highways as convex polygons that contain no known obstacles. Then a straight-line path connects any two points within a single highway region and planning reduces to finding a chain of overlapping highways that includes both the initial position and the goal position. On the other hand, if highways are arbitrary polygons and contain mapped obstacles, then planning a path within each highway becomes much more complex.

2.3 Control Loop

Given goal coordinates from the path planner, the control loop must direct the robot to that position. It is important that the control loop be reliable as well as complete. If it is not reliable, the robot will get "lost"; if it is not complete, the robot may fail to reach the goal point even if a path exists. Obviously, the control loop needs to interact with the physical robot, both to command changes in velocity and to receive sensor data. Furthermore, to achieve *reliable* motion, the control loop must model control uncertainty. Therefore, before we discuss the control loop itself we must define the control loop's representation of this uncertainty.

Definition 2 (Positional Uncertainty) *The positional uncertainty region $\mathcal{U}_t$ is defined as the region in which the robot is known to lie at time t.*

Note that there is nothing probabilistic about the uncertainty region—we know that the robot must lie within it. Also, note that the size of the region $\mathcal{U}$ will depend upon how well the robot can determine its current position. We assume that a robot has two general methods of position determination: by integrating its commanded velocity over time and by localizing based on sensory input and the geometric map. This means that the positional uncertainty is

```
while (¬Termination) {
        AcquireSensorData;
        if (DecideToLocalize)
            Localize;
        ComputeVelocity;
        CommandVelocity;
        UpdateUncertainty;
}
```

Figure 3: The general structure of the control loop

the result of two other types of uncertainty: control uncertainty (in the integration case) and sensory uncertainty (in the localization case).

Now that we have defined the uncertainty region, we can return to the discussion of the control loop. Figure 3 shows the high-level structure of the loop. We describe each component of the loop below.

`Termination` There are three possible ways for the loop to terminate:

1. The robot has achieved its goal. In the face of uncertainty, this means that $\mathcal{U}$ lies completely within the goal region (which encapsulates the goal point and allowable error).

2. The robot has become lost. This occurs when, in spite of efforts to localize based on sensory input, $\mathcal{U}$ remains so large that the robot cannot achieve the goal.

3. The robot has realized that there is no path to the goal. The control loop is constrained to travel only inside the current highway; therefore, this condition indicates that the robot has realized that an impassable obstacle is blocking the path to the goal.

`AcquireSensorData` In addition to acquiring sensor data from the robot level, it may be useful to fuse actual sensor data with "simulated" sensor data obtained by examining invisible, mapped obstacles in the geometric map.

Additionally, certain sensing processes such as vision may require too much processor time if done as part of a single-threaded control loop. Such sensor processes run asynchronously and `AcquireSensorData` would poll them as required.

`DecideToLocalize` This is the step in which the control loop must reason explicitly about the uncertainty region $\mathcal{U}$. This decision function tells the controller when it must re-localize and reduce the size of $\mathcal{U}$ in order to preserve goal reachability.

For example, if localizing is time-intensive, it would be appropriate to delay localization until the uncertainty region exceeds some threshold size. On the other hand, if localization is inexpensive, it would be beneficial to localize at regular intervals.

`ComputeVelocity` This step defines the system's control strategy, and could be implemented in many different ways. Its function is to combine obstacle avoidance with goal-directed behavior in order to calculate new velocities for the robot level motors. We require two guarantees: first, that the robot reach the goal when possible; and second, that it avoid contact with all sensed and mapped obstacles.

`UpdateUncertainty` As the robot moves, this routine extends $\mathcal{U}$ in accordance with the bounds placed on control uncertainty. This step is vital because it ensures the continuing validity of the uncertainty region, which must by definition always contain the robot's actual position.

2.4 Localizer

The success of the control loop depends on keeping the size of the positional uncertainty region $\mathcal{U}$ sufficiently small. Without the use of sensors, the size of $\mathcal{U}$ will, in general, only increase, since there is uncertainty in control. The role of the localizer is to use sensor data to compute a new region $\mathcal{U}_t'$ from the current region $\mathcal{U}_t$ and some set of sensor values. The hope is that $\mathcal{U}_t'$ will be smaller than $\mathcal{U}_t$, thus reducing the robot's positional uncertainty.

Note that the localizer may have internal state. In particular, this means that it may use a history of sensor values instead of a single instantaneous reading. The use of history can increase the effectiveness of the localizer by significantly decreasing the likelihood of a false localization.

3 The Vagabond Project

The Vagabond Project [Dugan and Nourbakhsh, 1993] was an effort to build a reliable outdoor navigator for the Stanford University Quadrangle. This outdoor arcade houses many of Stanford's departments and is composed of several walks that are flanked by regular pillars and sandstone walls.

Vagabond is a Nomad 100 mobile robot from Nomadic Technologies, Inc. It consists of a nonholonomic base which supports sixteen infrared sensors and sixteen sonar sensors. Its "brain" is an Apple Powerbook 170 that communicates with the sensor boards and motor controller through a serial link. The infrareds have an effective range of 0 to 15 inches while the sonars have an effective range of 15 to 150 inches.

3.1 Task Description

The Quad presents Vagabond with several great challenges. Many of the arcades are lined with six inch steps that would topple it, and, worse yet, the walks themselves have scattered potholes that are deep enough to trap it. In contrast to many forgiving office environments, the Quad allows Vagabond to actually destroy itself by mistaking its position. The dynamic character of this uncontrolled environment adds to the danger—at times bicyclists and pedestrians densely populate the walkways. Finally, direct sunlight in the Quad washes away infrared light, leaving Vagabond with sonar as its sole sensory input.

Given this very real environment, the task was to enable Vagabond to navigate successfully while avoiding the unmapped obstacles and the deadly steps. The final interface is precisely a navigator-level module. At the task level, the human provides initial position and orientation information and then supplies goal points through a graphical interface.

3.2 Implementation

Vagabond's map is a data structure with a polygonal description of every obstacle. The map differentiates visible from invisible obstacles. Overlaying this two-dimensional picture is a set of highways that are also represented as polygons. Figure 4 displays a portion of Vagabond's actual map. The filled polygons are mapped, visible obstacles while the unfilled polygons are mapped, invisible obstacles such as potholes. The shaded polygons depict the highways. Additionally, each highway has an associated speed limit that is based upon the general smoothness of its terrain.

Vagabond's path planner is an A* visibility graph search algorithm that treats both visible and invisible mapped obstacles as navigation points. The path planner finds the path with the fastest expected time of completion, based upon the top speed feature and the path length. The path planner then stores the path as a list of points to be achieved and sends the successive goal points to the control loop, waiting for success or failure and responding appropriately. In

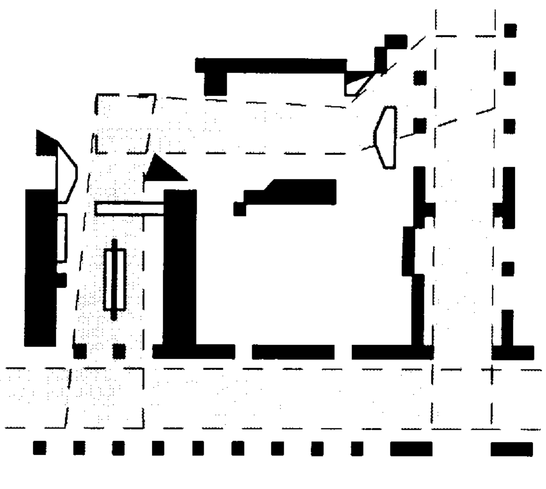


Figure 4: A section of the map of Stanford University Main Quadrangle, as used by Vagabond

the case of failure, the path planner recognizes that the goal point is not reachable from this highway, and so removes it from the map. It will then re-plan to find an alternate path to the task-specified goal point.

The control loop represents $\mathcal{U}$ as a rectangular region for the sake of computational efficiency. The `ComputeVelocity` routine employs a simple multi-level architecture with two behaviors: course maintenance and reactive obstacle avoidance. The course maintenance module resembles an aircraft course autopilot. It acts to reestablish the course and heading that define the line segment of travel between two successive subgoal points. The obstacle avoidance module modifies these ideal motion settings to avoid both sonar-detected obstacles and mapped invisible obstacles. Note that the obstacle avoidance module must ensure that the entire region $\mathcal{U}$ remains clear of any invisible obstacles on the geometric map.

The careful design of the interaction between these two modules is essential to preserving goal reachability as well as graceful behavior in the event of encountering an impassable obstacle. For instance, the desire to reestablish course should never override the refusal to allow $\mathcal{U}$ to overlap an invisible obstacle. However, intelligent obstacle avoidance demands more than a purely reactive decision system to avoid looping behavior.

The final ingredient of Vagabond's navigation system is the localization procedure. Localization is extremely time-intensive on Vagabond's hardware and is therefore minimized. The control loop only calls the localizer when the the size of $\mathcal{U}$ exceeds a threshold. The localizer has no state—it uses the current

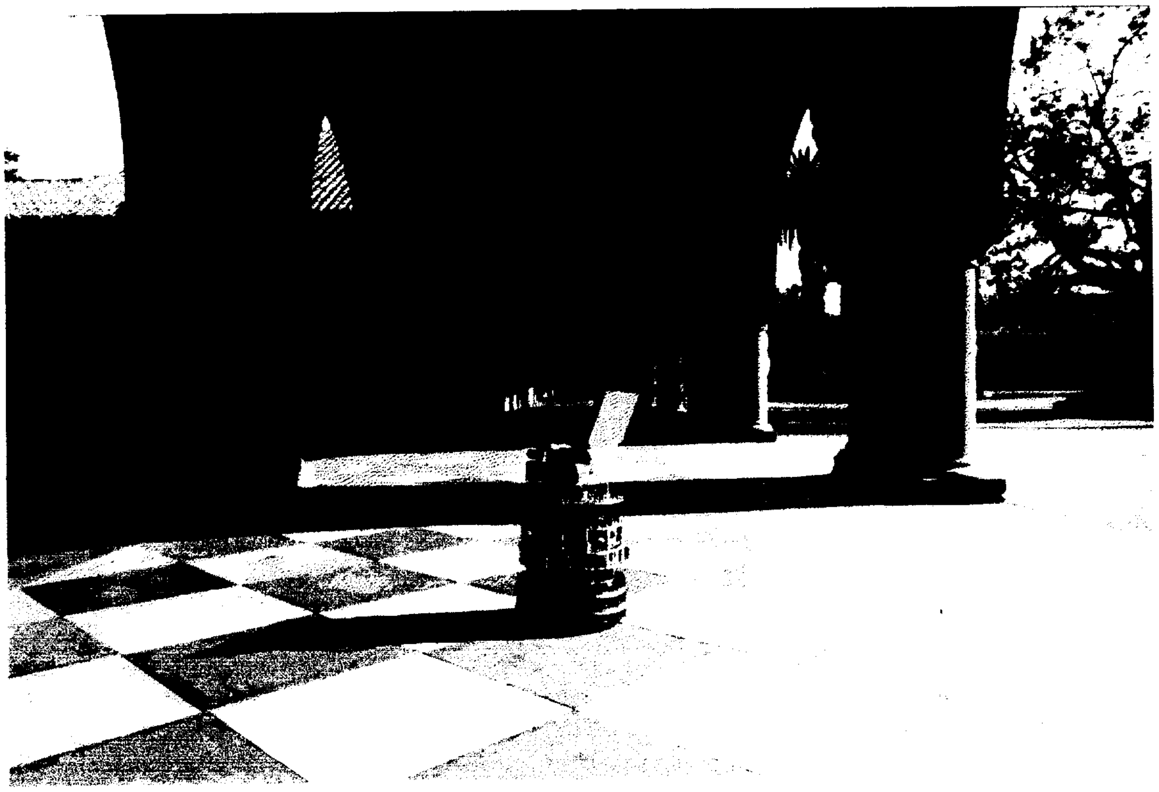


Figure 5: Vagabond navigating in the Stanford Main Quadrangle

instantaneous sensory input rather than a history of sensory data. It employs a deceptively simple scoring strategy that is surprisingly effective even in times of significant sensory occlusion (by people, bicycles, etc.). The key is the simple idea that any unexpectedly long real-world sonar value provides evidence for the elimination of a possible map position (sonars do not see through sandstone walls) while any unexpectedly short sonar value may be attributable to an occlusion by unmapped obstacles.

3.3 Results

One of the most desirable properties in a mobile robot is the ability to avoid self-destruction. For Vagabond, this meant always preserving its sense of position well enough to avoid the deadly steps. The architecture guarantees that no part of Vagabond's uncertainty region will intersect any mapped obstacle. Assuming that all steps are mapped (as they were), self-destruction could only occur after a false-positive localization. That is, Vagabond's localizer would have to localize to an incorrect location, thus violating the architectural assumption that the robot is always within $\mathcal{U}$.

Our goal was to produce a truly robust navigator. To this end, the entire development and testing process used the real world, never a simulator. We tested the final Vagabond system intensively in the Quad environment, both during quiet times (e.g. weekdays in summer) and in times of extremely dense traffic (e.g. between classes in the autumn). False localization occurred extremely infrequently during testing and never continued long enough to result in a deadly move. The only recurring cause of false localization involved onlookers who formed human walls parallel to and offset from the walls of the Quad. Sonar cannot differentiate such human walls from real walls. Happily, group dynamics seem to render human walls too transient to be a serious threat.

In contrast, Vagabond's most common failure resulted instead from an inability to localize successfully. This would eventually lead to an uncertainty region so large that it rendered any further movement impossible. In these cases, Vagabond would stop and return the "lost" termination condition to the task level. In our tests, this condition occurred in approximately 10% of all cases in which the user requested Vagabond to achieve a certain position on its map. Vagabond would reach the destination point and return success in the remaining 90% of the cases.

Vagabond moved at a slow walking pace (12 inches per second on average), typically covering distances of $\frac{1}{2}$ mile per task.

4 The SCIMMER Project

The SCIMMER[1] Project was organized to develop a successful entry for two contests at the AAAI Robotics Competition held in Washington, D.C. in July, 1993. The contests involved simple navigation tasks in contest arenas that simulated real-world conditions using gray office partitions, white boxes, and actual office furniture.

SCIMMER is a Nomad 200 robot from Nomadic Technologies, Inc. (Figure 6). It has a three-wheel synchronous drive non-holonomic base, on top of which is an independently rotating turret housing sensors and on-board computation. The sensors include 20 pressure-sensitive bumpers, 16 sonar sensors, 16 infrared sensors, a structured light vision system consisting of a laser and CCD camera, and a second CCD camera linked to a frame-grabber for vision processing. We ran all software on-board using a 386-based PC system.

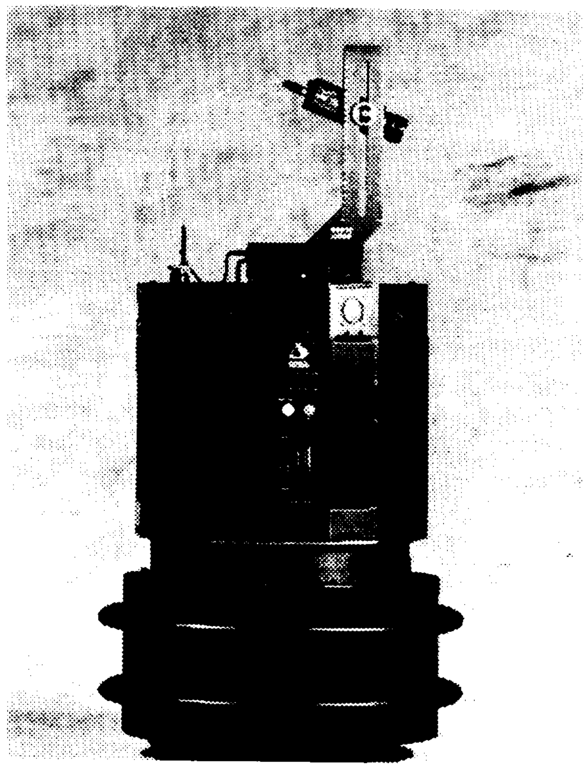


Figure 6: The Nomad 200 robot

4.1 Task Description

Contest I The environment was a large "warehouse" with an enclosed office at one end. SCIMMER's task was to escape from the inner office, then race to the far wall of the warehouse. The office contained typical office furniture (e.g. file cabinets and tables) while the warehouse was cluttered with white boxes.

Contest II The environment was a simulated office building with rooms and hallways connected in a fairly typical layout. White boxes were scattered around as obstacles. The goal of the contest was to find a coffee pot and deliver it to a specified room. At the start of the contest, the robot received a map of the office building (divided into quadrants), its starting quadrant, the quadrant containing the coffee pot, and the destination room for the coffee pot. Note that the robot begins the contest with an enormous amount of uncertainty as to its initial location, so a major part of this contest was the initial localization.

4.2 Implementation

Contest I required domain-dependent code for escaping the inner office, followed by an implementation of the control loop subsystem to reach the goal region. Readers interested in the control loop implementation are referred to [Balabanovic *et al.*, 1993]. Our Contest II entry provides a more complete implementation of the navigator level; this is the implementation we now describe.

SCIMMER's geometric map is a simple line drawing, with each line denoting a wall in the real world. There were no invisible but mapped obstacles (such as sharp drop-offs) in the environment. The highway map consists of both highways and nodes. Highways are polygons of free space (barring any unmapped obstacles). The nodes are simply just intersections between highways that provide task-level goal regions to facilitate movement between highways while simplifying path planning.

SCIMMER's planner uses a best-first search algorithm to find the shortest path from one node to another. The planner then feeds the control loop one node at a time. Because of the nature of the task, the planner does not re-plan if the control loop fails to achieve its subgoal. Instead, it returns impossible to the task planner. Consider the problem: we're trying to find a coffee pot in one quadrant of the map. There could be multiple rooms in that quadrant; if we find a blockade along the way, we might want to

[1] Sarah, Craig, Illah and Marko's Most Excellent Robot

change the order in which we visit those rooms. Since this is a high-level task decision, control must return to the task level.

The `ComputeVelocity` routine that combines these desires frequently commands the robot to move at the motor controller's top speed of 20 inches per second, as the contests were timed. Once the robot is within the goal region, the control loop exits to the planner, signalling **success**. In the case of failure, the control loop exits signalling **impossible** and the planner removes that highway from the map.

SCIMMER deals with positional uncertainty in a very simplified way. Upon reaching a goal node, the control loop decides whether it should localize by referring to the map, on which all nodes are marked either "localize" or "don't localize". We entered this information manually, basing our decisions upon the degree to which different nodes would be effective places to localize. For example, nodes in the middle of a long hallway would be very unreliable whereas nodes at an intersection of three of four highways would be promising.

SCIMMER's localization, as opposed to Vagabond's uses history. As it moves, it builds a bitmap representing the objects it has detected over time with its laser range-finder. The localizer uses a general shape matching algorithm to find the best match of this sensor history against a bitmap representing the known obstacles in the world. The shape-matching metric used is the Hausdorff distance, following the general algorithm presented in [Huttenlocher *et al.*, 1991].

Once again, we avoided the use of simulation altogether during the development of the SCIMMER contest entry. Success demanded fast, robust operation in the actual contest environment—therefore, we chose this environment as our development environment.

4.3 Results

Contest I SCIMMER achieved first place. It successfully avoided all obstacles and quickly followed a smooth path to the final goal.

Contest II SCIMMER was one of only two contestants to successfully localize itself at the start of the contest without assistance. It began to follow its plan to reach the projected location of the coffee pot, but an unfortunate operating system problem caused the robot to crash a short distance from that goal.

5 Limitations and Extensions

Clearly, there are domains to which this architecture simply does not apply. For instance, the problem of visually recognizing a coffee pot requires a specific solution that does not fit in our three-level decomposition. Indeed, any problem that does not require navigation between well-specified destination points will not benefit from our architecture.

A more serious limitation involves the explicit uncertainty region that the navigator level maintains. Although the control loops we have implemented based velocity decisions on the size of $\mathcal{U}$, among other parameters, neither of our systems incorporated reasoners that would move the robot exclusively to shrink $\mathcal{U}$. One can imagine a case in which the robot needs to move from A to B, yet the direct path is so sparse that the robot must first move from A to landmark C, where the size of $\mathcal{U}$ can be bounded, and then on to B. Our current implementations would fail in this situation because neither Vagabond nor SCIMMER's path planners account for the size of $\mathcal{U}$. A possible solution is to use a path planner that predicts the localizer's reliability at any given map location.

Another significant limitation of our architecture is that it fails to provide any mechanism allowing the robot to improve its performance over time by learning more specific information about its environment. The obvious solution to this deficiency is to allow the robot to modify its geometric map during navigation, thus attaining an increasingly accurate representation of its environment over time. In reality, this is an extremely complex issue that currently has no satisfying solution. Today's robotic sensory input is too imprecise and robotic common sense too undeveloped to allow a robot to make useful decisions concerning the transience of unexpected obstacles.

Finally, the robustness of any navigation system depends largely on the richness and reliability of its sensors. Sensors such as sonar transducers are useful in many situations, but their very nature renders them unable to detect many hazards (such as downward steps and narrow chair legs) that exist in the real world. It seems useful, then, to explore other types of sensors which do not suffer from these limitations.

One could imagine designing a specific "downward step sensor" using short-range proximity sensors or touch sensors trained on the floor. In fact, ground-level tactile sensors seem to complement sonar well, detecting many of the low-lying obstacles that otherwise evade detection.

Perhaps a better solution is an increased reliance on vision. Richer, more flexible sensing would improve the performance of our Navigator by allowing more precise localization and would allow us to reduce control error by receiving constant environmental feedback while moving. Our system makes it easy to incorporate such enhanced sensing, and we believe its development is vital to eventually building truly robust systems.

6 Related Work

Researchers from both the robotics and the artificial intelligence communities have been addressing the challenges of mobile robotics for some time. However, their approaches and the focus of their research have been quite different.

The robotics community has successfully addressed the challenges of many of the components of a robot architecture. Most of the subsystems we posit as part of the Real-World Navigator have been extensively researched. Crowley [1989] develops a localizer that uses ultrasonic range data to find a robot's position on the map. His approach involves an abstraction step in which the localizer extracts potential line segments out of the sonar data. Takeda and Latombe [1992] address the problem of path planning under the specific assumption that the executor will use sensory feedback to localize during path execution. Their sensory uncertainty field computation ascribes to each possible robot position a measure of the robot's ability to localize using sensory input at that position. For example, a corner would receive a much higher score than a featureless wall.

In contrast, the AI community has witnessed a recent spate of work on architectures for robotic agents. However, these agent architectures often blur the distinction between the task level and the navigator level. As a result, most AI robot architectures do not make the strong claim that is implicit in the Real-World Navigator: that the navigation component can be fixed across application domains. Instead, a common approach is to allow higher-level components to activate, deactivate or parameterize navigation processes. Recent examples include ATLANTIS [Gat, 1992], SSS [Connell, 1992] and [Saffiotti, 1993]. A further alternative is to compile beforehand a reactive structure that will execute a plan at run-time (again, navigation is neither a fixed component nor a necessary part of these structures). Examples include [Kaelbling and Rosenschein, 1989], [Schoppers, 1987] and [Nilsson, 1994].

Another important difference between the Real-World Navigator and many other current approaches is our need for a geometric map, enabling explicit maintenance of a positional uncertainty region. A popular alternative is to navigate using robust reactive routines such as wall-following and corridor-following, and to provide a connectivity map in terms of these motion primitives as well as high-level sensory primitives (e.g. T-junctions, doorways). This technique, which evolved from the subsumption architecture [Brooks, 1986], has been successfully demonstrated by [Gat, 1992] and [Connell, 1992]. The clear advantage of these systems is that they do not require a geometric map of the environment. However, the software is usually quite domain-dependent, and any change of domain requires a great deal of rewriting. In addition, many extensions (such as avoiding mapped, invisible obstacles) do not fit neatly into this framework. Finally, it is difficult to see how such a system would be able to effectively determine that it was lost.

Three projects at Stanford are worth noting here. The Logic Group formalizes the concept of planning with incomplete information and designs a framework in which an agent may act explicitly to decrease its uncertainty [Genesereth and Nourbakhsh, 1993]. Another project focuses on landmark-based navigation where assumptions about sensing and control within specific landmark regions are used to reduce planning to a polynomial-time problem [Lazanas and Latombe, 1993]. Finally, the AIbots project [Hayes-Roth *et al.*, 1993] addresses issues involving the interface to the task level by investigating the integration of a *cognitive level*, which is currently a BB1 blackboard system dealing with task planning and deadline management, with a *physical level* which includes a path planner and a navigator.

7 Conclusion

We have introduced an architecture for mobile robot control which addresses the problem of navigation. In addition to demonstrating robust behavior in dynamic, real-world situations, the two applications we have described show that the architecture is independent of the task domain, the environment and the robot platform.

Our belief is that the success of these applications is due not only to the design of the individual components, but also to the design of the architecture itself. This allows reuse of the architecture over many different tasks, its tested framework considerably decreasing the difficulty of building robust, general-purpose navigation software.

The Real-World Navigator provides a solid foundation on which we can build highly effective real-world mobile robot applications.

Acknowledgements

Benedict Dugan played a major role as one of the two developers on the Vagabond Project. Thanks are also due to Michael Genesereth, Barbara Hayes-Roth, Jean-Claude Latombe and Nils Nilsson for their helpful comments and suggestions. Finally, we would also like to thank David Zhu, Jim Slater and John Slater at Nomadic Technologies for all their help in times of need.

References

[Balabanovic *et al.*, 1993] Marko Balabanovic, Craig Becker, Erann Gat, Steven Goodridge, David Hinkle, Ken Jung, Sarah Morse, Illah Nourbakhsh, Harsh Patlapalli, Reid Simmons, and David Van Vactor. The winning robots from the 1993 robot competition. *AI Magazine*, Winter (In Print) 1993.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[Connell, 1992] Jonathan H. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992.

[Crowley, 1989] James Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989.

[Dugan and Nourbakhsh, 1993] Benedict Dugan and Illah Nourbakhsh. Vagabond: A demonstration of autonomous, robust, outdoor navigation. In *Video Proceedings of the IEEE International Conference on Robotics and Automation*, 1993.

[Gat, 1992] Erann Gat. Integrating planning and reacting in a heterogenous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the 10th National Conference on Artificial Intelligence*, 1992.

[Genesereth and Nourbakhsh, 1993] Michael Genesereth and Illah Nourbakhsh. Time-saving tips for problem solving with incomplete information. In *Proceedings of the 11th National Conference on Artificial Intelligence*, 1993.

[Hayes-Roth *et al.*, 1993] Barbara Hayes-Roth, Philippe Lalanda, Philippe Morignot, Karl Pfleger, and Marko Balabanovic. Plans and behavior in intelligent agents. Technical Report KSL-93-43, Stanford University Knowledge Systems Laboratory, 1993.

[Huttenlocher *et al.*, 1991] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. Technical Report CUCS TR 91-1121, Cornell University Department of Computer Science, 1991.

[Kaelbling and Rosenschein, 1989] Leslie Pack Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6(1–2), June 1989.

[Latombe, 1991] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[Lazanas and Latombe, 1993] Anthony Lazanas and Jean-Claude Latombe. Ladmark-based robot motion planning. In C. Laugier, editor, *Geometric Reasoning for Perception and Action*, pages 69–83. Springer-Verlag, Berlin, 1993.

[Nilsson, 1994] Nils J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, To Appear 1994.

[Saffiotti, 1993] Alessandro Saffiotti. Some notes on the integration of planning and reactivity in autonomous mobile robots. In *AAAI Spring Symposium on Automated Planning*, 1993.

[Schoppers, 1987] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 1039–1046. International Joint Committe on Artificial Intelligence, 1987.

[Takeda and Latombe, 1992] H. Takeda and J.C. Latombe. Sensory uncertainty field for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992.

6051

N94- 30556

AIAA-94-1205-CP

A STREAMLINED SOFTWARE ENVIRONMENT FOR SITUATED SKILLS

Sophia T. Yu, Marc G. Slack and David P. Miller
The MITRE Washington AI Technical Center, Mail Stop Z401
7525 Colshire Drive, McLean, Virginia 22102-3481
syu or slack or dmiller @starbase.mitre.org (703) 883-7738

p. 7

Abstract

This paper documents a powerful set of software tools used for developing situated skills. These situated skills form the reactive level of a three-tiered intelligent agent architecture under development at the MITRE Corporation. The architecture is designed to allow these skills to be manipulated by a task level engine which is monitoring the current situation and selecting skills necessary for the current task. The idea is to coordinate the dynamic activations and deactivations of these situated skills in order to configure the reactive layer for the task at hand. The heart of the skills environment is a data flow mechanism which pipelines the currently active skills for execution. A front end graphical interface serves as a debugging facility during skill development and testing. We are able to integrate skills developed in different languages into the skills environment. The power of the skills environment lies in the amount of time it saves for the programmer to develop code for the reactive layer of a robot.

1 Introduction

Within the short history of robotics research, many different approaches have been proposed for creating the intelligent component of an autonomous entity. The majority of them were considered unsatisfactory for developing an R2-D2-like robot. For instance, the traditional school of thought, grounded on real-world modeling and planning, was criticized for the discrepancy between the real-world and the computer model. Although more recent approaches based on simple reactivity algorithms produced surprisingly intelligent appearing robot behaviors [2], many argue that these robots are incapable of complex tasks [9].

Despite the absence of an R2-D2 legacy, the two approaches lay the foundation for a middle ground approach. This approach incorporates both a deliberative and a reactive component. Most people will agree that an architecture that includes both components seems to be a sensible way to build the robot brain. After all, human beings appear to have both reactive and deliberative faculties [5]. Deliberation allows the robot to make plans and predictions, and reactivity allows the robot to respond effectively to uncertainties. However, the tough question is how to put together a system with both a deliberative and reactive component?

A number of systems of this nature have been proposed and tested. Rosenschien and Kaelbling developed the situated automata theory for robot control [6]. The authors describe a system which would allow high-level description of the environment to be translated into situation appropriate low-level control activities. Arkin's AURA builds an accurate global model of the world and passes this information to a vector summing control layer [1]. Payton and Rosenblatt's architecture has four layers: task-level planning, global path planning, local path planning, and reflexive control [8]. The first three layers are equivalent to a traditional planning layer. The fourth layer consists of numerous reactive experts whose influence on the actuators is arbitrated by a central module. Erann Gat proposes a three-tiered architecture ATLANTIS [4]. In this architecture, a sequencing layer integrates and pipelines the deliberative and the reactive functions, which ultimately control the robot.

The MITRE autonomous systems laboratory also participates in the pursuit of an intelligent robot system characterized by deliberation and reactivity [10]. Our system is similar to an architecture originally proposed by Firby [3] and further developed by Gat [4]. The MITRE intelligent agent architecture (MIAA) consists of three interacting layers. The deliberative layer makes high-level decisions which require reasoning about resource and time constraints. The reactive layer consists of situated skills that react to the situation at hand. Finally, a sequencer which acts as a temporal and syntactic differential between the reactive and deliberative layers, decomposes planning structures into the appropriate activations and deactivations of the robot's skills (See Figure 1).

The design of the three-tiered architecture is based on two important concepts: heterogeneity and asynchrony. The system is heterogeneous in that it is composed of components that are structurally different, e.g., time-consuming decision-making modules versus instant response reactive modules. The system is asynchronous in that deliberative, sequencing, and reactive modules are executed in parallel and communicating to each other via asynchronous message events. This guarantees that reactive modules respond to the situation at hand in a timely fashion and that an adequate amount of time is allocated for deliberative processes.

This paper describes the design of an environment for constructing situated skills (the term appears in [10]) which form the lowest layer of MIAA. The environment must establish communication channels with the sequencer in order to accept commands from the deliberative layer. The environment must also be able to streamline the skills so that each skill provides a timely response for given sensor data. Moreover, the environment must provide a means for integrating the individual skills into a dynamically reconfigurable library of robotic skills.

In this paper, we shall answer explicitly three questions in sequential order. What are the necessary requirements for engineering a desirable skill development environment? How is the environment implemented? Finally, why does the implemented skills environment constitute a valuable set of tools?

2 Requirements

There are two sets of requirements for engineering a desirable software environment for the situated skills. The first set of requirements address the specifications for an infrastructure capable of controlling skills. The second set of requirements focus on software engineering related issues. This second set of requirements is especially important since robotic systems are becoming increasingly complex, raising information management as a central issue.

In order to better understand the first set of requirements, it is necessary to examine what is required of the reactive layer separately from the rest of the architecture or more accurately from the sequencing layer. The main function of the sequencing layer is to provide runtime situation-driven execution [3]; its job is to transform a set of partially ordered plan steps from the deliberative layer into the necessary set of skill activations and deactivations to accomplish the high level plan for the current situation. To achieve this, the reactive layer must accommodate the demands of the sequencing layer. For example, the reactive layer must provide perceptual information continuously and without delay. In addition, the reactive layer must provide a mechanism that allows the sequencing layer to tailor the behavior of a skill according to different situations.

The skills environment divides naturally into two main parts: a skill library and a mechanism which pipelines the skills and interfaces with the sequencer. The skill library provides the basic functionality of the instantiated autonomous agent. Skills include not only the computational elements of perception but the interfaces with the robot's hardware. This latter feature provides a mechanism for the interchangeability of physical devices and portability of the architecture. Each skill also provides conditional mechanisms which allow the sequencer to adapt to the situations at hand. Finally, each individual skill provides control switches for activation and deactivation.

A number of temporal and dependency requirements dictate the design of the mechanism used to control the skills. First, all components of the architecture must have access to relevant perceptual information and have the capability to control system actuators. Secondly, the perceptual information should be available continuously and that at every instance in time, actuation commands are assigned to external actuators for given sensor data. Thirdly, there should be an asynchronous and distributed execution of skills. In other words, it should be possible to execute skills in parallel as long as there is no contention of resources among the parallel skills. Lastly, unlike the deliberative functions, these situated skills should have low-commit and fast response time since real-time performance is necessary.

The second set of requirements for the skills environment is derived from the vantage point of a good software engineering project. One of the main goals of a good software project is to increase the number of users of the program. The design strategy taken is to encapsulate the details of the skills environment, therefore those without knowledge of the architecture should be able to program skills tailored to the domain of their robot. Another design requirement taken into consideration is the reuse of old software written in different languages. The implementation phase for configuring a robot can be substantially cut if existing code can be reused as part of the skills library and thus integrated into the overall control paradigm. This requirement can be satisfied by implementing standard encapsulations

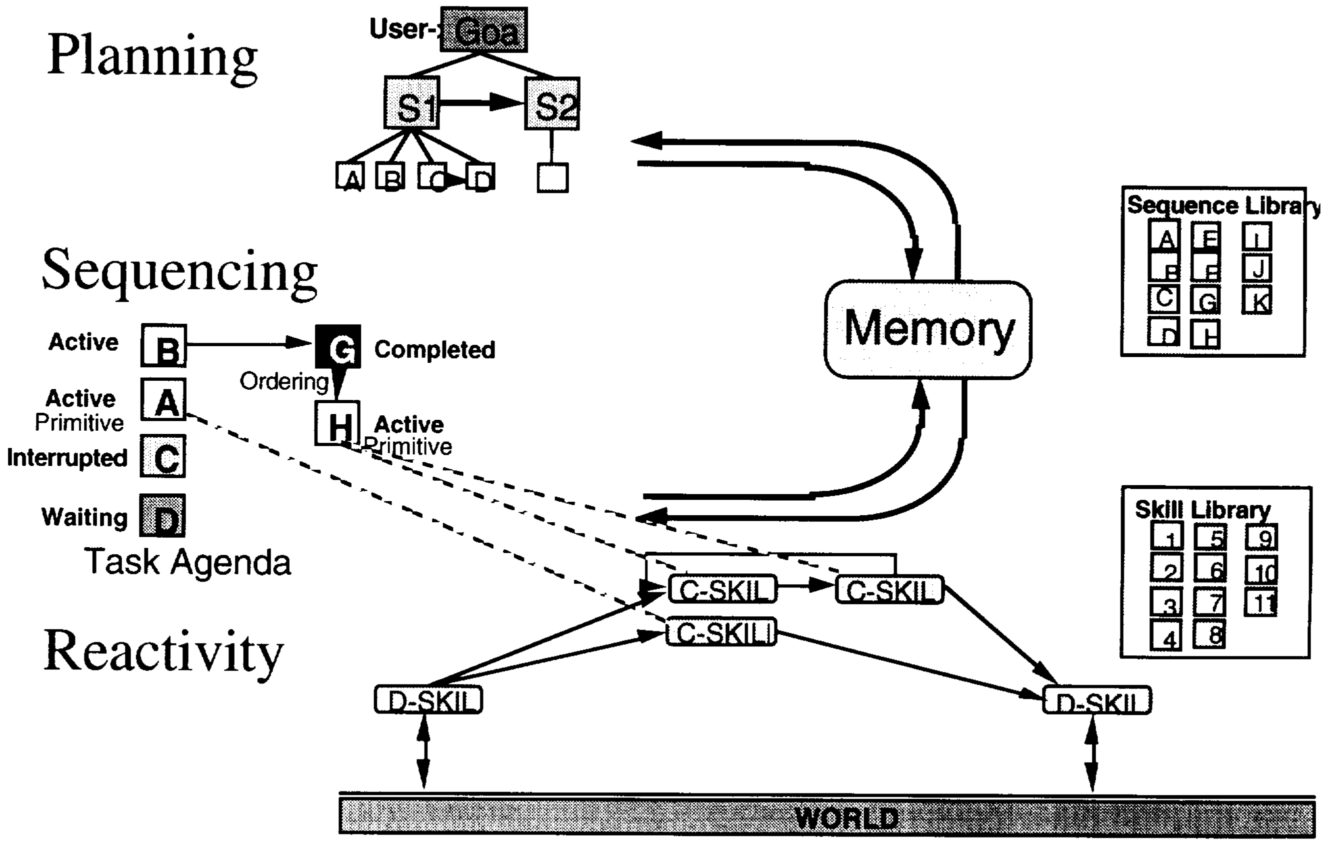


Figure 1: The flow of information through MIAA.

and interfaces for skills. In addition, there should be support mechanisms to facilitate the painful process of debugging. Lastly, it should be easy for users of the skills system to activate and deactivate skills manually, so that skills can be debugged and tested in isolation prior to their integration into the rest of the architecture.

3 Structure

This section discusses the design of the skills environment. We took an object oriented approach in designing the skills environment. Essentially, the skills environment is composed of two classes of structures: skill and skill manager. Each class is associated with objects and functions, and two instances of a class have access to the same class objects and functions. We will discuss the two classes of skills by focusing on their associated objects and functions.

3.1 Skills

The skills object class defines a set of structures which support the reactive modules. These structures serve as encapsulations of the reactive modules, to provide a standard interface to all of the reactive modules. Additionally, this interface includes structures for textual or graphical display of parameters, parameter logging and debugging purposes.

The objects for the skill class are the common data structures among every reactive module. For example, the input and output data structures of reactive modules are objects. A good portion of objects are used to support the scheduling of skill operations. There are also objects which are used for interfacing the skills with the sequencer. A priority slot is also available for resolving conflicts among skills in contention for resources. In addition, memory locations are allocated for recording parameter values and for displaying information associated with a skill.

A skill's parameters are important data objects which deserve special attention. Each skill has its own unique set of parameters. These parameters are singled out because they play an important role in bringing about situation-driven execution. These parameters are used and modified by the sequencing layer to tailor the behavior of a skill to the situation at hand. For instance, the sequencing layer may increase the priority value of a skill to give it temporal precedence over other skills. Another example of a skill parameter is the `safe-distance` parameter in the `runaway` skill. When an instantiated robot encounters an obstacle-dense region, the sequencer may decide to decrease the value of the `safe-distance` parameter so that the robot is able to pass through the obstacle region, then restore the parameter later when the robot is out of the congested region.

Because each skill will generally have a different set of inputs and outputs, there are numerous functions that are automatically generated by the skill object class when a specific skill is instantiated. This allows the environment to be skill independent and frees the skill designer from the concerns of interfacing the skill to the other skills in the skill library. The designer must only be concerned with the input and output requirements and the necessary computation; all interfacing issues are automatically handled by the development environment.

The skill class is further categorized into two sets of skills, those which are purely computational (or cskills) and those which interface with devices (or dskills).

3.1.1 Cskills

Cskills are skills which obtain their inputs from other skills and perform a computational transform on the inputs and pass the transformed values to other skills. Cskills can be operated either synchronously or asynchronously. In the synchronous case, a cskill will run its computational transform whenever it is given a new set of inputs, blocking until the computation has completed and a new set of outputs has been generated. In the asynchronous case, the cskill will continuously perform its transform on the currently available inputs and will respond immediately (like dskills) with the latest answer. The asynchronous cskills are especially useful when cskills are being executed on a distributed computer network.

3.1.2 Dskills

A dskill serves as an interface to a physical device (see Figure 2). This interface brings actuation commands to and obtains sensory data from the physical device. Rather than performing a computational transform on the inputs of the skill, a dskill buffers its inputs and provides a mechanism for sending those inputs to a device. A dskill will also buffer the latest sensor readings from the device. These sensor readings are provided as the users output from the dskill. The reason behind the creation of a separate skill class is to

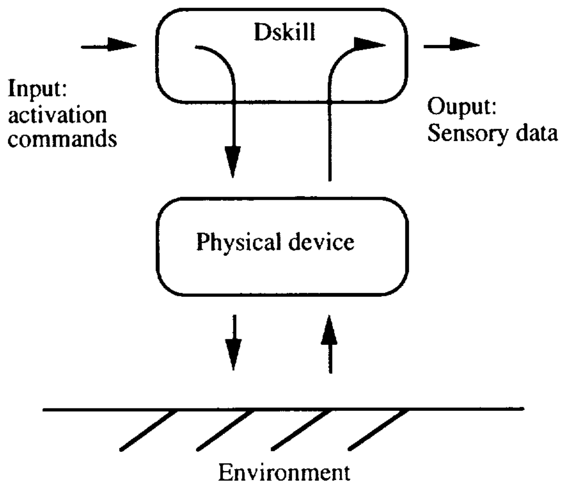


Figure 2: Information flow of a dskill.

provide the developer with mechanisms for handling the delay in communications associated with device drivers. Thus a dskill will always respond immediately and will not block for communications events as device interfacing is handled in a separate asynchronous process.

3.2 Skill Manager

Instances of the skill manager class are responsible for the timely activation and deactivation of reactive modules. They provide inter-skill communications as well as communications with the sequencing layer. The paradigm used for scheduling the execution of the skills is a data flow mechanism.

3.2.1 Data Flow

The objects of the skill manager class are mainly data structures supporting the data flow mechanism. For instance, there are allocations that store a list of instantiated dskills and cskills. A slot is reserved for counting the number of cycles the data flow mechanism has executed. There is also a state slot which regulates the sequence of function invocation in a cycle. In addition, there are data structures that keep track of the activation and deactivation requests made by the sequencing system.

There are two critical functions central in understanding the workings of the data flow. They are `do-periodic-step` and `update-record`. The `do-periodic-step` is the driving function behind the data flow. Once this function is invoked, an infinite loop starts. During each cycle of the loop, two functions are called in sequential order:

`do-next-skill`, and `cleanup`. The first function creates an environment for the executing the set of activated skills, and the last function destroys the environment created preparing it for the next cycle.

The order in which the individual skills are executed during a cycle depends essentially on input readiness. The update-record function is instrumental in preparing for the readiness of skills. Every time a skill produces an output, the `update-record` function checks these output data structures against the input data structures to skills yet to be executed. If there is at least a partial match in data structure, the `update-record` function avails the matched data structures to the yet to be executed skills. When all the inputs to a skill are available to it, the skill is ready and is executed the next time that the skill is considered for execution.

3.2.2 External Control

For the system to be of use there must be a mechanism for determining which set of skills should be active at any moment of time or the overhead of this paradigm is wasted as one could have simply hacked up a large C file to provide the necessary utility. However, as mentioned earlier, there are different syntax and semantics required to construct the different levels of abstraction necessary for constructing autonomous agents. In our system, the sequencing layer is assumed to handle the process of activation and deactivation of skills. The reason for handling skill activity in the sequencing layer is that the sequencer is maintaining an explicit representation of the robot's current situation (e.g., navigating down a hall, opening a door, etc.). It is beyond the scope of any individual skill in the currently active network of skills to be able to interpret the context and decide how the current sensor information should be interpreted with respect to the current task.

To support the use of a sequencer, the skill manager maintains an asynchronous communications link through which requests are made. The skill manager handles not only requests for activation and deactivation of skills, but also requests for state monitoring, parameterization, and value queries. The ability of the reactive layer to take initiate monitoring events is critical to any multi-layered architecture as the sequencer knows which information is critical to the task yet the skill level represents the only location where high frequency information can be captured. For example, the sequencing system could setup a monitor asking the skill manager to send an asynchronous event back when the robot's front sonar reads less than 10 inches. Because the sequencing and skill layers of the architecture operate in parallel, such information is too transient for the sequencing layer to capture directly yet there is insufficient information in the skill layer to determine that the information means that the robot has reached the ticket counter. In a similar fashion, the skill manager also allows the sequencer to make direct queries of the state of the skills in the reactive layer, thus allowing the sequencer to obtain instantaneous primitive value readings (e.g., is barcode 3 currently visible?). Lastly, the skill manager allows the sequencer to set the parameters of individual skills. This allows the skills to be dynamically configured for the situation at hand.

4 Implementation

The current skills development and execution environment is implemented within the Macintosh operating system. The structures discussed in the last section are implemented with the Common Lisp Object System (CLOS) application for two reasons. First, it is easy to realize the object oriented features of the skills environment; the CLOS package has constructs that accommodate class objects, functions, and instantiations. Secondly, the CLOS package has an easy to program graphics package. The graphics package is used to allow the skills programmer to control the operations of the skills environment with ease. The graphical interface is implemented mainly for debugging purposes. The next paragraph explains the implementation of the graphical interface in more detail.

Since there are two classes of structures, two types of graphical interface were designed and implemented: one for activating the individual skills and one for activating the data flow mechanism. The graphical interface for the skills class has mouse-clicking buttons for activation, parameter logging, and textual display. In addition, it has optional buttons for changing the values of skill parameters. The graphical interface for the skill manager class has a button for activating the data flow mechanism and a variable set of buttons for individually activating the set of instantiated skills. Note that these graphical interfaces are objects of the skill and skill manager class. These buttons of the graphical interface allow the skills programmer to activate a skill or a set of skills and to observe runtime execution results.

To utilize the environment, the job of a skills programmer is fairly simple since most of the inter-skill communication and graphical interfacing issues are handled by the generic skill object. To implement a skill, the skills programmer needs to specify only three things: input, output, and computational body. These three items must be properly placed into the template provided by the structures within the skills class.

The skills environment is set up in a way to permit the implementation of a multilingual skills library. This is possible for two reasons. The encapsulated and modular design of the skills environment keeps the differences among skills within the skills themselves, while the uniform and standard features of the skills environment provides a direct way for communications among the disparate skills. We have implemented skill construction facilities to allow a programmer to create skills written from C, C++, Pascal, Assembler, LISP, and REX [7]. To program a skill in a language other than LISP requires slightly more work. In addition to specifying the three things in a different language, the input and output data structures must be specified in LISP so that memory allocations are made properly in the skills environment.

The implemented skills are fairly easy to debug with the help of the graphics interface. To debug a skill, the first step is to instantiate that skill. Next, the user may want to change the values of the skill parameters to the desired values by clicking on the parameter button(s). Pressing the activation button will start the execution of the skill. The user is able to view the runtime parameters of individual skills by clicking on buttons on the graphical interface window. Clicking on the *show data* button invokes a corresponding window which is capable of displaying text. The *show input* and *show output* buttons forces the input and output of skills to be displayed on the secondary window, and the *verbose* button allows the display of any print statements generated internally to the user's skill. In order to debug a skill within the context of a set of skills, it is necessary to activate the data flow mechanism. The first step in this process is to instantiate a set of needed skills. The next step is to instantiate a skill manager for pipelining the instantiated skills. Pressing the run button on the skill manager window will start the data flow mechanism. Runtime results of activated skills can be viewed on the secondary windows of the instantiated skills.

5 A Valuable Set of Tools

The power of the skills environment lies in the amount of time it saves for the skills programmer. During virtually every stage of the skills development cycle, time is conserved. For instance, learning time is shortened. The programmer does not need to have extensive knowledge of the skills environment to program a skill and integrate it into the skills environment. The details of the skills environment are encapsulated; the object-oriented constructs of the skills environment essentially provide templates for programming and instantiating reactive modules.

Moreover, programming time is significantly reduced in two ways. First, the skills programmer does not need to worry about interfacing with other skills or other components of the architecture. Recall, the programmer needs to specify only three things to program a skill. Secondly, the skills environment allows the reuse of existing code. This capability is valuable since rewriting code such as the robot's inverse kinematics is a task one would like to do only once.

In addition, debugging time is decreased. As mentioned in the last section, there are a number of debugging facilities available. The graphic interface allows the easy operation of these facilities. Also, the modular decomposition of the skills allow the individual skills to be debugged in isolation or in the context of other skills.

The maintenance of the skills library also becomes less time-consuming. Modifying the skill library to adapt to the changing capabilities of the instantiated autonomous agent is quite straightforward. Since the interface to physical devices is encapsulated in dskills, adding, deleting, or replacing dskills is all that is needed to adapt to a change in physical devices. The components of the rest of the skills environment remain unaltered.

6 Proposed Experiment

Before concluding the paper, we relate our experience of programming a set of skills for our proposed experiment with MIAA. We offer this to demonstrate the time conserving way of programming skills. The proposed experiment is for our Denning robot which must deliver a message to our department head.

The robot wakes up in its humble abode: the autonomous systems laboratory in the basement of the building. It wanders around the laboratory, avoiding obstacles and looking for a door to exit. Once the door is found, it exits the door. Since our department head's office is on the fourth floor, the robot must find the elevator first. It directs itself to the elevator using a combination of hallway following, door detection and intersection detection. Once the elevator doors are found, the robot pushes the elevator button and waits for the elevator. When the elevator comes it must determine which of the four elevator cars actually arrived. Upon entering the elevator the robot must push the button for the fourth floor. When the elevator stops on the correct floor, the robot exits.

It directs itself down the corridor to the department head's office and delivers the message.

The set of behaviors described above can be accomplished with a standard set of situated skills and a specialized set of task specific skills. Some examples from the standard set are obstacle avoidance, wall following, wandering, object tracking, and object homing. An example from the specialized skill set is a module that allows the robot to push an elevator button.

As an example of how the sequencer coordinates the situated skills to bring about the set of behaviors described above consider the task of exiting the autonomous systems laboratory. Five skills are key in creating this behavior: obstacle avoidance, wandering, barcode tracking, locating the position of the door, and position homing. A barcode is placed in the vicinity of the door so that the robot can reliably recognize the door's general location. The sequencer first activates wandering, obstacle avoidance, and barcode tracking, so that the robot wanders around the laboratory, avoiding obstacles and looking for the barcode associated with the door. Once the location of the barcode is found, the sequencer activates the barcode homing module, commanding the robot to move in front of the barcode. Finally, after the robot moves to the vicinity of the door, the sequencer activates a module that computes the necessary position clues for isolating the door opening and driving the robot through the door.

From this simple example, you can see the utility of being able to activate and deactivate skills depending on which aspect of the overall task the robot is currently working on. For example, it makes little sense to spend valuable computation time identifying the door opening until the robot is in the vicinity of the door. We are collecting metrical information, to provide evidence of the utility of the explicit sequencing of skills verses the implicit sequencing of skills typical of more ad hoc reactive techniques. It is our hypothesis that as the size of the class of tasks within a given domain increases the utility of taking a more structured and engineered approach to design of robotic intelligence will clearly win out over the "purely" reactive techniques.

7 Final Words

The skills environment is a powerful technology for a number of reasons. It abstracts the skill developer from the details of communications protocols and graphical user interface issues which the environment provides. A person writing code for a sequencer has only to concern themselves with which skills are needed and can ignore all of the inter-skill communications issues as these are handled by the data flow mechanism of the skill manager. We believe that by providing methodology to the creation and use of reactive modules that the work in reactive control of robots can move out of the ad hoc creation of task-specific demonstrations into the world of assisting in the solution of real-world problems.

References

[1] R.C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *International Journal of Robotics and Autonomous Systems*, 6(1-2):105–122, 1990.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14-23, March 1986.

[3] R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds.* PhD thesis, Yale University Department of Computer Science, January 1989. see Technical Report 672.

[4] E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.* PhD thesis, Virginia Polytechnic Institute Department of Computer Science, April 1991.

[5] P.N. Johnson-Laird. Mental models in cognitive science. *Perspectives on cognitive science*, pages 147–191, 1981.

[6] L. Kaelbling and S. Rosenschein. Action and planning in embeded agents. *Robotics and Autonomous Systems*, 6:35–48, 1990.

[7] L. P. Kaelbling. Rex: A symbolic language for the design of parallel implementation of embedded systems. In *Proceedings of the AIAA Conference on Computers in Aerospace*, 1987.

[8] D. W. Payton. An architecture for reflexive autonomous vehicle control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1838–1845, April 1986.

[9] M. G. Slack. *Situationally Driven Local Navigation for Mobile Robots.* Department of computer science, Virginia Polytechnic Institute, April 1990. also published as Jet Propulsion Laboratory Publication 90-17.

[10] M. G. Slack. Sequencing formally defined reactions for robotic activity: Integrating RAPS and GAPPS. In *Proceedings of the SPIE Conference on Sensor Fusion*, November 1992.

~~:~~-63
207064
60552
AIAA-94-1206-CP

SITUATIONAL REACTION AND PLANNING

p. 6

N94-30557

John Yen and Nathan Pfluger
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

Abstract

One problem faced in designing an autonomous mobile robot system is that there are many parameters of the system to define and optimize. While these parameters can be obtained for any given situation, determining what the parameters should be in all situations is difficult. The usual solution is to give the system general parameters that work in all situations, but this does not help the robot to perform its best in a dynamic environment. Our approach is to develop a higher level situation analysis module that adjusts the parameters by analyzing the goals and history of sensor readings. By allowing the robot to change the system parameters based on its judgement of the situation, the robot will be able to better adapt to a wider set of possible situations. We use fuzzy logic in our implementation to reduce the number of basic situations the controller has to recognize. For example, a situation may be 60 percent open and 40 percent corridor, causing the optimal parameters to be somewhere between the optimal settings for the two extreme situations.

Introduction

The design and implementation of autonomous mobile robot planning and control systems will allow robots to handle tasks that are very dangerous or impossible for a human controlled system to handle correctly. Instances of these tasks are the autonomous rover system for the exploration of Mars, and the battlefield tank controller, where ECM may make it impossible to remote control tanks in enemy territory.

On problem faced in designing an autonomous mobile robot system is that there are many parameters of the system to define and optimize. Some of these parameters include maximum safe speed, how near obstacles can be without being threats, and how far the robot can safely project its destination. While these parameters can be obtained for any given situation, determining what the parameters should be in all situations is difficult. The usual solution is to give the system general parameters that work in all situations, but this does not help the robot to perform its best in a dynamic environment. The problem of finding parameters is further complicated if the robot is able to perform many missions, like exploring or transporting, which each require the robot to behave differently in identical environments.

Our approach is to develop a higher level situation analysis module that adjusts the parameters by analyzing the goals and history of sensor readings. By allowing the robot to change the system parameters based on its judgement of the situation, the robot will be able to better adapt to a wider set of possible situations. We are currently implementing the situation analysis module using fuzzy if-then rules. Use of fuzzy logic allows us to specify less base situation and to combine these situations into more types of situation easier than a standard logic based system.

The rest of this paper is organized as follows. First we will give a brief overview of fuzzy logic and fuzzy control. We will then give an overview of intelligent systems. We will then cover situation recognition and its potential benefits. We will then discuss our testbed and an implementation of situation recognition in it. We will then give some results from simulations of our mobile robot controller.

Background

In this section, We will first give a brief overview of fuzzy logic and the use of fuzzy logic in decision making and control. Afterward we will discuss the development of intelligent systems.

Fuzzy Logic

Motivated by the observation that many concepts in the real world do not have well defined sharp boundaries, Lotfi A. Zadeh developed fuzzy set theory that generalizes classical set theory to allow objects to take partial membership in vague concepts (i.e. fuzzy

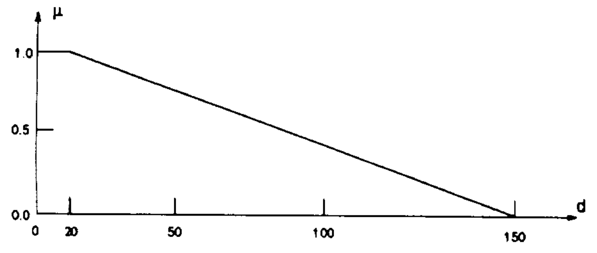


Figure 1: A Fuzzy Set Representing the Concept of Near

sets).[10] The degree an object belongs to a fuzzy set, which is a real number between 0 and 1, is called the *membership value* in the set. The meaning of a fuzzy set, is thus characterized by a *membership function* that maps elements of a universe of discourse to their corresponding membership values.

Figure 1 shows the membership function of the fuzzy set **NEAR** in the context of mobile robot navigation control. In this Figure, d represents the distance of the closest obstacle detected by a sensor, and μ represents the membership value in the fuzzy set **NEAR**. As the Figure depicts, an object that is 15 units away has a full membership in **NEAR**, while one that is 50 units away has a membership value of 0.8. Capturing vague concepts such as **NEAR** using fuzzy sets can improve the robustness of a navigation control system in the presence of sensor noise, because noise in the sensor data can only slightly change the membership degree in **NEAR** and therefore affects the final control command in a minor way.

Based on fuzzy set theory, fuzzy logic generalizes modus ponens in classical logic to allow a conclusion to be drawn from a fuzzy if-then rule when the rule's antecedent is partially satisfied. The antecedent of a fuzzy rule is usually a boolean combination of fuzzy propositions in the form of "x is A" where A is a fuzzy set. The strength of the conclusion is calculated based on the degree to which the antecedent is satisfied. A fuzzy logic controller uses a set of fuzzy if-then rules to capture the relationship (i.e. the control law) between the observed variables and the controlled variables. n each control cycle, all fuzzy rules are fired and combined to obtain a fuzzy conclusion for each control variable. Each fuzzy conclusion is then *defuzzified*, resulting in a final crisp control command. An overview of a fuzzy logic controller and its applications can be found in the work by C.C. Lee.[3, 4]

Intelligent Systems

We are presently delving into the use of fuzzy logic in the implementation of intelligent systems. A definition of an intelligent system from a paper by Albus[1] is that an intelligent system must at least have the ability to sense the environment, make decisions and to control actions. Higher levels of intelligence may require the recognition of objects, the storage of knowledge for future uses, and the ability to reason how actions will affect the future.

All of these functions will be needed by a system that wishes to act in complex dynamic environments effectively. An example of such an application is an autonomous mobile robot system. The lower level abilities of sensing and control are used by the robot to quickly sense and avoid obstacles in the path. The higher level abilities of recognition and projection are needed to allow the robot to adjust itself to any environment.

We have used fuzzy logic to implement a behaviorial based system that is able to use sensors to control the robot to follow a given path while avoiding obstacles in the path, a discussion of which is given in the next section. We are currently working on using the ability to recognize elements of the environment and to reason about how those elements will effect the robot in order to compute the most effective parameters to use in any given situation.

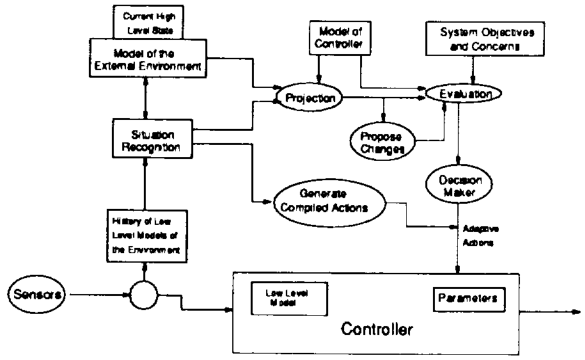


Figure 2: High Level Model of Situation Recognition and Adaptation

Situation Recognition

In this section a general architecture for including situation recognition is given. We will then overview our method for situation recognition and reaction of the system to recognized changes in the situation.

General Architecture

The general architecture we have adopted for situation recognition and reaction is given in Figure 2. The main concept is to develop a system independent of the controller that has the ability to adapt the controller based on its perception of the situation, i.e. a metalevel controller. The architecture has one set of inputs, the sensors, and produces a set of adaptive actions for the controller.

The sensors are fed into a situation recognition architecture along with a high level model of the environment. This purpose of this module is to examine the sensor histories and the changes in the high level model

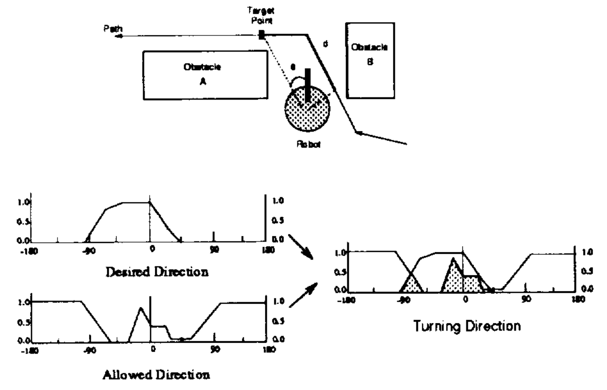


Figure 3: Desired and Allowed Directions

to determine if there has been a significant change in the environment to warrant adaptive action for the controller. How this module operates is given in the next subsection.

Once a new situation has been recognized, there are two possible methods of handling the change. The first is to directly generate precompiled actions for those changes. These actions are then fed to the controller for adapting to the current situation.

The other method of adaption is to use a form of projection to find the possible effects of the change in environment. If the effects are large enough, then changes need to be proposed and the new system is projected till a satisfactory projection is found.

The architecture given has not been fully implemented, and only represents the final system we want. At present the method of adaption is to use a set of precompiled actions to change the controller based on the perceived situation.

Testbed

We are currently implementing the situation recognition architecture on an autonomous mobile robot control system. In this section, we will give a brief overview of autonomous mobile robot systems and the fuzzy logic based behavioral architecture which we are currently using.

The basic problem of autonomous mobile robot path planning and control is to navigate safely to one or several target locations. The problem can be further complicated by other considerations such as deadlines for reaching those locations, safety considerations of paths, reactiveness to emergent situations and uncertainty about the environment. There are several approaches that accomplish this goal. We will concentrate on a behavioral implementation that uses fuzzy logic to merge sensor readings with path information to determine the final control command each cycle.[8, 9]

The approach we have taken uses fuzzy logic to describe the desired and allowed direction of travel, see Figure 3. The algorithm first determines the target point by projecting along the path given. It then uses fuzzy logic to broaden it into the desired direction. Next, the inputs from sonar sensors fixed around the body are fuzzified and combined to form the allowed direction. These two concepts are then combined to form a fuzzy control command that describes what the robot should do. After using Centroid of Largest Area (CLA) defuzzification,[7] a control command for the robot can be found. For a more detailed discussion, please see Yen and Pfluger.[8, 9]

The above method, as is the case with most sophisticated control systems, has a problem in specifying the control parameters to handle all situations. Some of the parameters whose optimality can be dependent on the environment include distance to target point, maximum speed fcr both going straight and turning, and *nearness* of objects for both forward and side sensors. When the robot was first programmed a set of values was taken such that robot would be able to work in any environment, i.e. very conservative values. The goal of this research is to improve the performance of the robot by changing the values of the control parameters in reaction to perceived changes in the environment.

Implementation of Situation Recognition

The first step is to determine what are the salient features of the situation that we want to recognize. At present we attempt to find three features:

1. *Openness:* This gives a general measure of the number of obstacles, both seen and expected.
2. *Path Information Strength:* Is the path along a road or a corridor? Should the path be followed religiously or just be taken as a possible path.
3. *Degree of Exploration:* Are we exploring the environment or traversing it? Or maybe a little of both.

Determining the amount of openness can be determined directly from the sensors. The method we are currently using is to use the allowed direction computation from the controller. This fuzzy set is determined by finding the nearness of obstacles and combining the results. By taking the fraction of the area that is allowed, we get a good measure of openness. This fraction can be further modified by the current values of nearness, i.e. if the nearness of obstacles is tight then the openness of the environment is less open.

The path information strength can be found both from the sensors and the path planning module. If the area is not open then the path strength should be stronger. If the goal of the robot is to explore, then the path strength should be less. The degree of certainty that the path is safe is passed from the planning module and influences this value. By using fuzzy rules to combine these and other concerns, a relative strength of the path can be found.

Finally, the degree of exploration can be determined from two sources. First, if the overall goal of the robot is specified then the degree of exploration is that. This can be modified by the number of undetermined obstacles found and the degree of uncertainty the robot has about the current map.

There is unfortunately a lot of interdependence of each of the situation features on each other. The path information strength, for instance, depends on both the openness and the degree of exploration. To handle this interdependence, this feature can be calculated after the others are finished.

For example, the rules to determine the Openness are:

- If Degree of Allowable is *High* and the Map indicates the number of obstacles is *Very Low* then the Openness is *Very High*.
- If Degree of Allowable is *High* and the Map indicates the number of obstacles is *Medium* then the Openness is *Medium-High*.

These are only some of the rules. They require that the Allowable direction be analyzed to determine the percent of objects that the sensors see, and for the map to do a count of the number of obstacles in the area. Other rules for Degree of Exploration and Path Strength require this measure of Openness including more information from the map and the goals to get the final results.

Using Precompiled Changes

Once the situation has been evaluated by the situation recognition module, we need to output what changes are needed to best adapt to that situation. We use another fuzzy logic rule base to react to the recognized change in situation provided by the situation recognition module. This rule base uses the results to reason about what the values of the parameters should be based on previous tests that have been run on the robot in each of the given situations. At present the system returns the adaption values of three system parameters:

1. *Target Distance:* how far along the path should the target point be
2. *Nearness of Sensors:* what should be considered near? Are there different values for the forward and side sensors?
3. *Maximum Speed of the Robot:* This value is for both the value when going straight and for making turns.

The target distance depends on all three features. The amount of openness is a guide to how far the robot may safely look ahead. The stronger the path information strength the less the robot can look ahead. Finally, in exploration, the path is usually just a guideline that the robot should stay near.

How near obstacles are for the sensors determines the sensitivity of the sensors. If they are too sensitive, the robot cannot travel in enclosed places. If they are too weak, then the robot cannot travel at large speeds

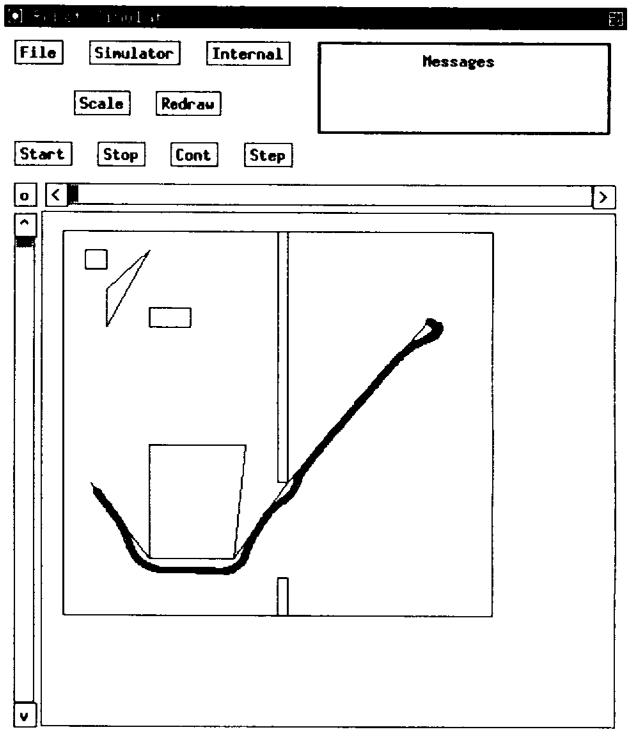


Figure 4: Outdoor Environment using Indoor Parameters

since it cannot sense obstacles soon enough to avoid them. This shows that the main overriding factor for nearness is the openness of the environment.

Finally the maximum speed of the robot is determined by combining the amount of openness with the goal of the robot. If the robot is exploring the robot will go slower to get more detailed scans. If the robot is traversing, it wants to go faster. If the robot is in an open area it can travel faster than in a less open area.

There is also some interaction between these parameter adjustments. The amount of nearness is influenced by the speed of the robot. The robot needs to be able to detect things further away as it goes faster. The robot also needs to project further ahead and anticipate corners better in open environments.

An example rule for generating an action is:

- If Environment is *Very Open* and Degree of Exploration is *Low* then Maximum Speed is *Very High*.
- If Maximum Speed is *Very High* and Degree of Exploration is *Low* then Target Distance is *Far* and Nearness is *Loose*.

The interdependance of the attributes can be seen in these rules.

Results

We have been working on implementing the situation recognition module. Figures 4-7 show two situations,

C-4

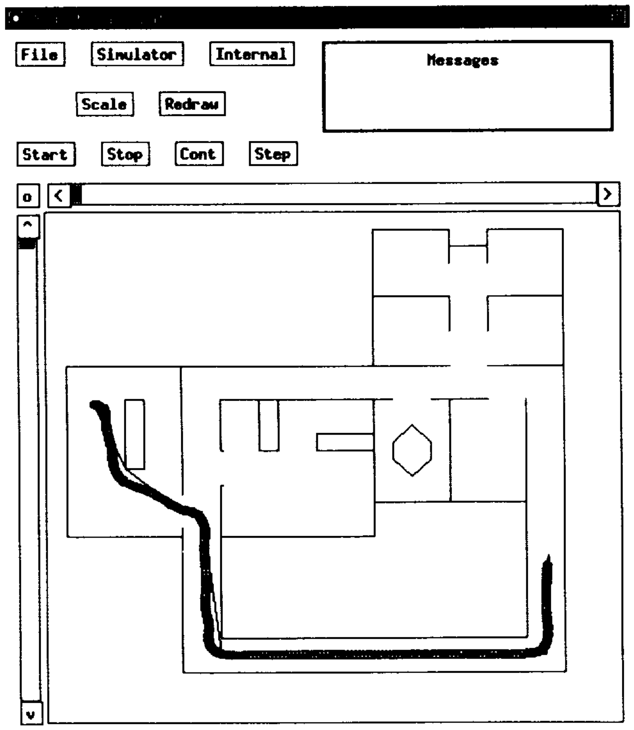
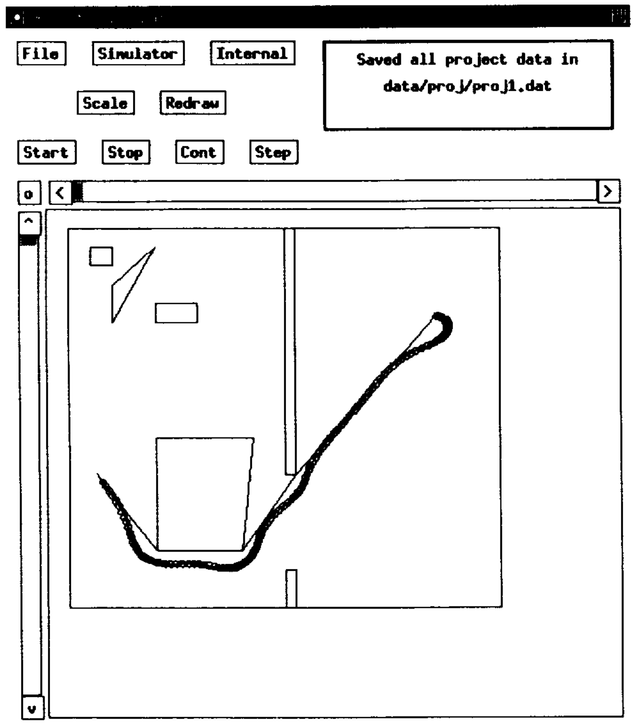


Figure 5: Indoor Environment using Indoor Parameters



Saved all project data in data/proj/proj1.dat

Figure 6: Outdoor Environment using Outdoor Parameters

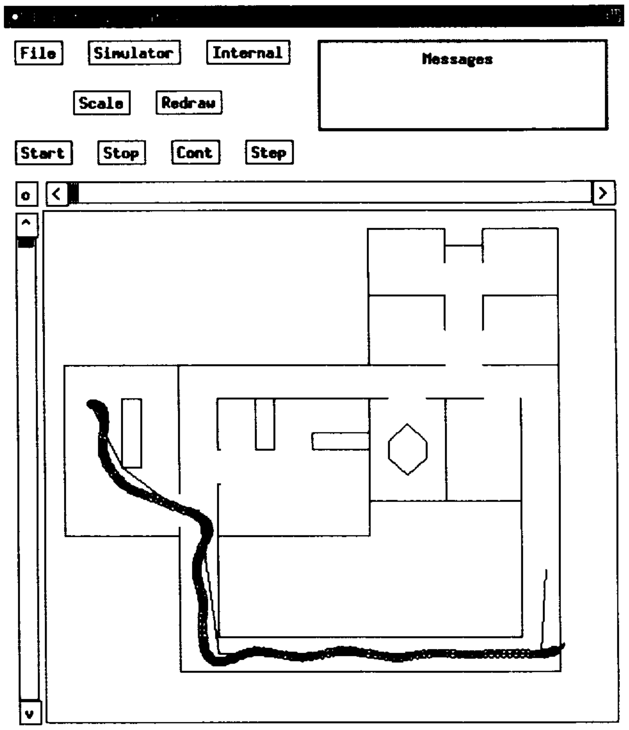


Figure 7: Indoor Environment using Outdoor Parameters

one an outdoor scene, the other using a floor plan of our lab. The figures show two different sets of parameters to show how changing the parameters can make the following of the path more efficient.

In Figures 4 and 5, the indoor parameters where used, meaning that the maximum speed was set to 10, the target distance to 50, and *nearness* to 30. The nearness is the point in Figure 1 where the membership stops being one and begins declining towards zero. In Figures 6 and 7, the outdoor parameters were used, i.e. the maximum speed was set to 17, the target distance to 70, and *nearness* to 60.

Important features of these figures include the fact that using the outdoor parameters, the indoor path could not be completed, while outdoors in a more open environment, the path may be slightly longer, but the time to complete the path was 150 steps, as opposed to 193 steps for the indoor paramenters, an improvement of 129 percent in the time needed to complete the task.

These runs were done using the same parameters throughout the run. In the future, the robot will be able to change parameters dynamically, based on the degree of openness and changing goals, as described in the section on situation recognition. This will allow the robot to be even more efficient, going faster when it can and slowing down in less open environments.

Adding Learning

We have recently been exploring the possibility of adding learning to the situation recognition and the action generation modules. Both of these modules use fuzzy logic rule bases to make decisions concerning the environment and how to react to it. While fuzzy logic rules are in general easy to create, there can be some problems.

The first problem is in scope. As the number of variables increase and the range of values they can take on becomes large, it becomes almost impossible to specify results for all possible combinations of values. The second problem occurs because rule bases are specified on the most important features of the problem. What if there are exceptions that can only be detected using variables not used in the fuzzy rule decision making process. Finally there is the problem that the rule base is created by an expert who is unsure of his reasoning and may give sub-optimal actions in a given situation.

We are currently working on learning in fuzzy rule based systems by using case based learning. The case based learning system adds cases to the rule base to overcome the problems created by using a static fuzzy rule base. We are researching ways to add cases so that their true motivation, whether it be as a novel case that should act as a rule itself, or as an exception which changes a small section of a rules scope.

Conclusions

As shown in the results section, there is a need to adjust the parameters based on the situation the mobile robot system is in. The current method of identifying the environment and determining the parameters is a start, but a more dynamic approach is needed. We are working on implementing the full system so that it can be run dynamically with the system. This will allow the robot to speed up in open areas while slowing down to safer speeds in more crowded environments.

References

[1] J. S. Albus. "Outline for a Theory of Intelligent". *IEEE Trans. Systems, Man, and Cybernetics*, 21(3):473–509, May/June 1990.

[2] B. Kosko. *Neural Networks and Fuzzy Systems*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[3] C.C. Lee. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I". *IEEE Trans. Systems, Man, and Cybernetics*, 20:404–418, Mar 1990.

[4] C.C. Lee. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part II". *IEEE Trans. Systems, Man, and Cybernetics*, 20:419–435, Mar 1990.

[5] D. W. Payton. "An Architecture for Reflexive Autonomous Vehicle Control". In *IEEE Conference on Robotics and Automation*, pages 1838–1845, 1986.

[6] D.W. Payton, J.K. Rosenblatt, and D.M. Keirsey. "Plan Guided Reaction". *IEEE Trans. Systems, Man, and Cybernetics*, 20:1370–1382, Nov 1990.

[7] N. Pfluger, J. Yen, and R. Langari. A Defuzzification Strategy for a Fuzzy Logic Controller Employing Prohibitive Information in Command Formulation. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 717–723, San Diego, CA, March 1992.

[8] J. Yen and N. Pfluger. "Designing an Adaptive Path Execution System". In *Proceedings of the IEEE/SMC Conference*, pages 1459–1464, Charlottesville, VA, October 1991.

[9] J. Yen and N. Pfluger. "Path Planning and Execution Using Fuzzy Logic". In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, pages 1691–1698, New Orleans, LA, August 1991.

[10] L. A. Zadeh. "Fuzzy Sets". *Inform. Contr.*, 8(3):338–353, June 1965.

6053

AIAA-94-1207-CP

AUTONOMOUS MOBILE ROBOT TEAMS

N94- 30558

532-63
[illegible]
p. 9

Arvin Agah & George A. Bekey

Institute for Robotics and Intelligent Systems
Computer Science Department
University of Southern California
Los Angeles, California

Abstract

This paper describes autonomous mobile robot teams, performing tasks in unstructured environments. The behavior and the intelligence of the group is distributed, and the system does not include a central command base or leader. The novel concept of the Tropism-Based Cognitive Architecture is introduced, which is used by the robots in order to produce behavior, transforming their sensory information to proper action. The results of a number of simulation experiments are presented. These experiments include worlds where the robot teams must locate, decompose, and gather objects, and defend themselves against hostile predators, while navigating around stationary and mobile obstacles.

1. Introduction

Teams of robots can be used in a wide variety of applications. Deploying a number of robots in an unknown environment can greatly increase the extent of the area covered for the research mission of planetary explorations, or surveillance of buildings and structures. A team of robots can provide the robustness required in critical missions, where the break down of one unit should not jeopardize the entire mission. The coordination of groups of robots allows them to perform tasks that are too large to be completed by one robot.

A team of robots could function as a centralized group, where the robot that act as the leader can assign sub-tasks to the other robots and monitor and manage the group. In distributed teams, the robots cooperate and perform the task without a leader. Although each type of cooperation has its own advantages, the leadership requirements have the disadvantages of requiring the leader to communicate with all the other robots. Such communications could be costly, and the entire system can come to a halt in the case of the leader's failure to function properly.

This paper describes the study of behavior of a group of distributed robots, surviving and performing tasks in an unstructured environment. We have termed the study of robot team behaviors as **Sociorobotics**[1]. In addition to the existence of stationary and mobile obstacles in the world, hostile entities (predators) exit in the world. These predators are mobile and capable of attacking and immobilizing the robots. The world also includes objects of interest to the robots. These objects could be picked up and collected by the robots. If the objects are too large, they must be first decomposed by the robots, before they can be collected. The robots' tasks mainly consist of locating and collecting small objects, locating and decomposing large objects, and locating and attacking predators. These actions are referred to as gather, decompose, and defend, respectively. An example of such tasks is shown in Figure 1. These tasks are performed in the world, while navigating around stationary and mobile obstacles.

Each robot senses and acts upon the world, using a novel architecture, termed **Tropism-Based Cognitive Architecture**. This architecture is based on the tropisms of the robot, i.e., its likes and dislikes. Such architecture transforms the robot's sensing of the world to potential appropriate actions. The cognitive architecture is tested using simulated robots in an artificial world. This world is similar in its characteristics to an actual world, and the facts and rules of the world are maintained and enforced by the artificial world simulator. The simulator generates an animated world, where the effects of changes inflicted upon the world can be dynamically viewed. In addition, the simulator includes an user-interface for the setting up of the experiments.

The Tropism-Based cognitive architecture enables the robots to survive and function in an unknown world. The desirable feature of such architecture is in its simplicity. Other approaches to cognitive architectures for intelligent systems include the hierarchical structure of intelligence[2], Subsumption type architectures based on augmented finite state automata[9,15], neural network based systems[3,5], synthetic psychology[8], reflex action control[6], and approaches to achieving general intelligence[12,14]. Examples of multiple robot systems include the schema-based navigation[4], subsumption-based systems[16], cellular robotic systems[11,13], artificial life systems[10], and swarm intelligence[7].

This paper is organized into six sections. The cognitive architecture is defined in section 2. Certain concepts in sociorobotics are discussed in section 3. Section 4 includes the description of the world, and the world simulator. Section 5 presents a number of performed experiments, and their results. Section 6 contains the conclusions.

2. Tropism-Based Cognitive Architecture

The cognitive architecture of each robot is based on the transformation of its sensory information to an action. The architecture will use the concepts of positive and negative tropism[17]. An agent's likes and dislikes will form its perceptions and, therefore, will result in its actions in the Tropism-Based Cognitive Architecture.

The sensing of the entities in the world includes the entity type and the state of the entity. For instance the entity that is sensed could be a predator and the state of the predator could be 'active'. Denoting the set of entities, the set of entity states, the set of robot's actions, and the tropism values by $\{\varepsilon_i\}$, $\{\sigma_i\}$, $\{\alpha_i\}$, and $\{\tau_i\}$, respectively, with $0 \le \tau_i \le \tau_{max}$, the tropism values can be represented by a set of relations. In each relation, given the entity and the state of the entity, the robot's action, and the tropism value will be determined.

$$\{ (\varepsilon, \sigma) \rightarrow (\alpha, \tau) \} \qquad (1)$$

In the above example the associated action could be for the robot to attack the predator. The larger the magnitude of the tropism value, the more likely it is for the robot to perform the action.

Once a robot performs a sensory sweep of its surroundings (available sensory area), the set of the tropism values are checked for any matching entity and entity state. For all the matched cases, the selection and the corresponding tropism value is marked. The selection of one action from the chosen set is done by using a biased roulette wheel. Each potential action is allocated a space on the wheel proportional to its tropism values. Then a random selection is made on the roulette wheel, choosing the action. Figure 2 depicts the roulette wheel, where the selection based on the wheel results in the action that is to be performed by the robot. Although currently the tropism values are preset for each robot, work is in progress to have the robots dynamically set these values based on their experiences, i.e., learn. This work is carried out under the research effort called *Project Sophia*.

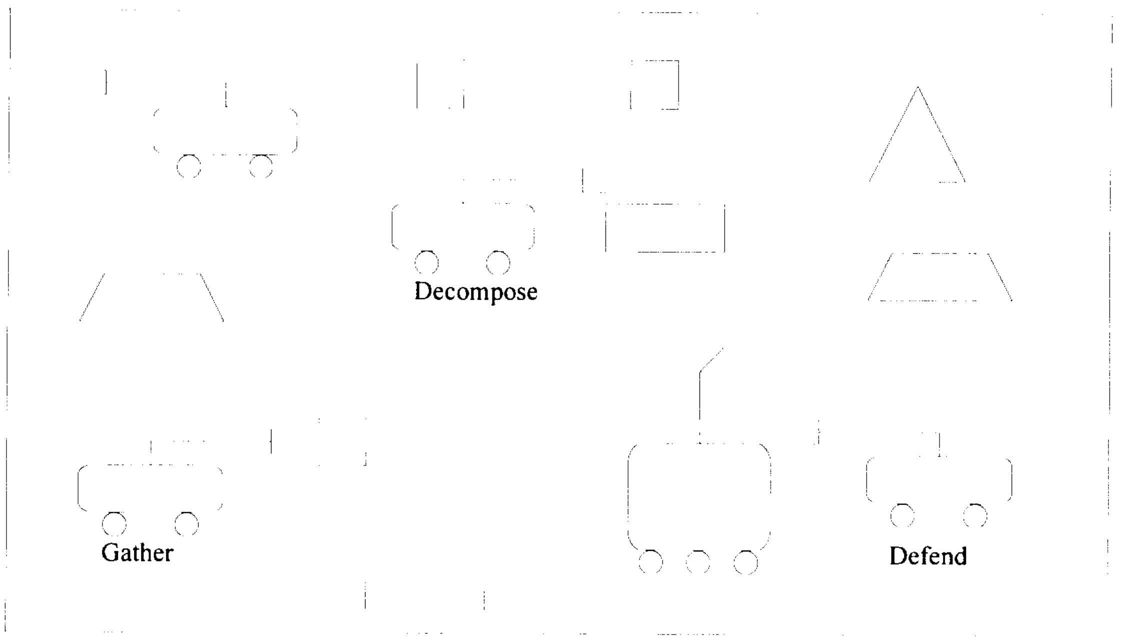


Figure 1: The robot team performing tasks in the world.

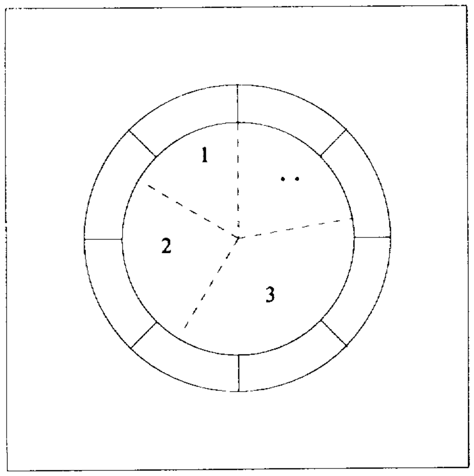


Figure 2: The biased roulette wheel for tropism values.

3. Sociorobotics

The study of the behavior of societies of robots is termed sociorobotics. Teams of robots are to survive and perform certain tasks in the world. The performance of the robots as a team is considered, in addition to the individual performance of each robot. Issues pertaining to the task performance of groups of robots are studied as parts of sociorobotics, including: task conditions necessitating a group, environmental factors influencing the group, appropriate group sizes, leadership and its form in a group, structure of a group (including the mixture of specialized versus generalist group members), behavior patterns of the group members, enhancements of group performance, and communication and its format. These concepts are analogous to those of sociobiology[18].

The parameters that are considered in the study of the behavior of robot teams include: the total elapsed time, the total energy consumption of all the robots, and the difference between the current and the final (desired) world status. The goal is to minimize these values, and by defining the total fitness of the team of the robots as the inverse of these values, the goal is to maximize the fitness function, denoted by Φ. Given the set of entities Y, the set of artificial world rules Γ, the time T, the initial and the desired worlds W, W^F, the set of all robots Ψ, and the fitness multipliers φ_T, φ_E, and φ_W, the fitness function Φ must be maximized.

$$\left(Y, \Gamma, W, W^F, \Psi, \varphi_T, \varphi_E, \varphi_W\right) \Rightarrow \max_R [\Phi(R, \varphi_T, \varphi_E, \varphi_W)] \tag{2}$$

Where φ_T, $_E$, and $_W$ are fitness multipliers that correspond to the strength of the corresponding time, energy consumption, and world status difference, respectively. Additionally, the role of these multipliers is to convert the units to a scalar. The multiplier φ_T is of inverse time units and the multiplier φ_E is in inverse energy units. The multiplier φ_W is a scalar. The matrix function $\|\ldots\|_2$ is the Euclidean norm of the matrix. The addition of 1 to the denominator is to prevent division by 0. The robot society is considered to be more fit for higher values of the function Φ.

$$\Phi(R, \varphi_T, \varphi_E, \varphi_W) = \frac{\varphi_T}{1+T} + \frac{\varphi_E}{1 + \sum_{t=1}^{T} \sum_{\psi=1}^{\rho} \Delta E} + \frac{\varphi_W}{1 + \|W^F - W^T\|_2} \tag{3}$$

4. World & World Simulator

The world within which the team of robots reside includes a number of different entity types. These include large and small objects, manipulated by the robots, stationary and mobile obstacles, and mobile predators. The robot is capable of performing action on these entities, as presented in Table 1.

Entity	Action
Space	Move
Obstacle	None
Base	Enter / Exit
Robot	None
Other	None
Predator	Attack
Small Object	Decompose
Large Object	Pick / Place

Table 1: Entities and the corresponding robot actions.

The world is a two-dimensional space, subdivided into individual blocks that could be occupied by an entity of any type. The center of the world is considered to be the home base of the robots, and the world is divided into eight zones, namely, North, South, East, West, North-

East, North-West, South-East, and South-West. Figure 3 displays the divided zones of the world.

A robot is capable of sensing and performing action on any of its eight surrounding blocks. The world blocks are enumerated as a two-dimensional matrix, with a row and a column specifying each block. Table 2 includes the block row and columns for the neighboring blocks.

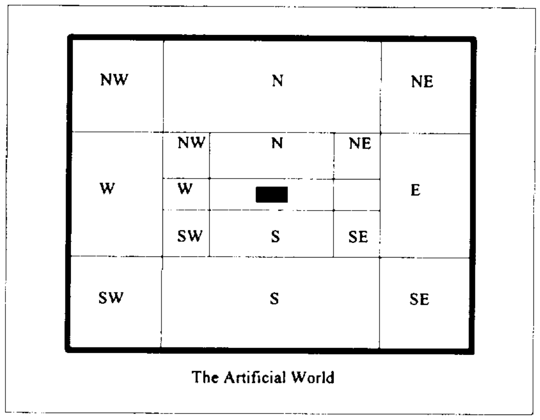


Figure 3: Divided zones of the world.

Direction	Row	Column
N	row - 1	Column
NW	row - 1	Column - 1
W	row	Column - 1
SW	row + 1	Column - 1
S	row + 1	Column
SE	row + 1	Column + 1
E	row	Column + 1
NE	row - 1	Column + 1

Table 2: Row and columns for the eight directions.

The number of zones accessible by a robot decreases once a robot is at the bordering block of the world. By convention it is assumed that the robot is surrounded by stationary obstacles in such cases. For example, once on the very corner of the world, three of the eight blocks are considered to be obstacles. The neighboring blocks of a robot are shown in Figure 4.

The animated display of the world is done using the world simulator. The display of the entities in the world is done in different color schemes. For instance predators are shown in light red, when active. Inactive predators are shown in dark red. Robots are displayed in purple, obstacle in gray and black and objects in blue and green. The world simulator includes the following modules:

• A graphics program for the animated display of the world and its entities.

• A user interface for the administrator to setup and conduct experiments.

• Algorithms to enforce the artificial realities.

• Algorithms to keep track of entity states, including the energy consumption of robots (Each robot consumes energy as it performs a task, proportional to the type of task).

• Algorithms to simulate the cognitive architecture of the robots and to decide the operations of the robots in the world.

The system is implemented on a 80486-based IBM-compatible computers, running Windows 3.1 operating system. The programming is done entirely in C programming language, including the algorithms, the user interface and the graphics. The program is compiled using Quick-C for Windows.

Figures 9 displays the setup screen for an experiment. Figure 10 shows an instance of the world and its robots and other entities. The displayed information include the population of the robots, the total time of the experiment, the total energy consumed by the robots, and the performance of the robot team in terms of gathering, decomposing and defending. The entity at the center of the world is the home base and the larger entities are the large objects.

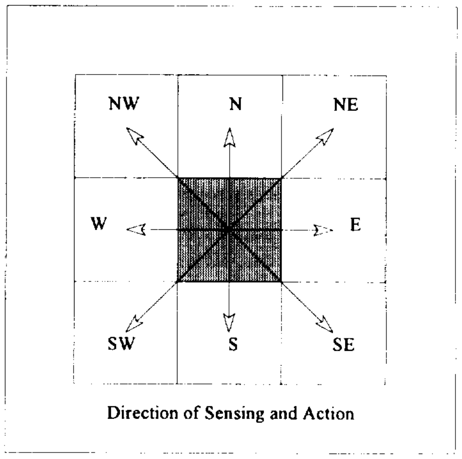


Figure 4: Accessible zones for sensing and action.

5. Experiments

Two series of experiments were performed with tames of robots controlled using the Tropism-Based cognitive architecture, using the world simulator. In the first series of experiments, the effects of the stationary and mobile obstacles on the energy consumption and performance of the robots were studied. In the second series of experiments, the effects of the robot team size on the energy consumption and performance were investigated.

All the experiments in the first series included 10 robots, 0 predators, 20 large objects, and 20 small objects. The total time of each experiment was 1200 simulation time unit, and the maximum performance achievable was 80 units. The numbers of stationary and mobile obstacles were equal, and their total varied from 0 to 128 obstacles. The graphs for the performance and energy consumption are plotted in Figures 5 and 6, respectively. In all graphs the actual data is in drawn using a solid, thick line, and the fitted curve is done using a dashed, thin line.

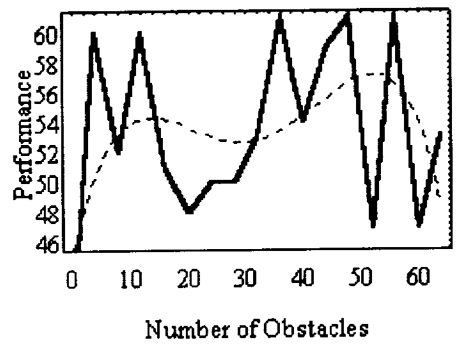


Figure 5: Team performance vs. obstacle density.

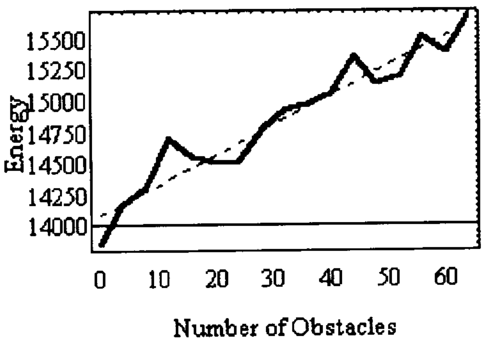


Figure 6: Team energy consumption vs. obstacle density.

As shown, as the number of obstacles increases, the performance increases, although the randomness in the placement of the objects and stationary obstacle results in the non-smoothness of the curve, which is fitted using a degree four polynomial. The energy consumption is linear with respect to the obstacle density, as obstacles result in more energy for the maneuvering.

The experiments in the second series included robot populations from size 0 to teams of 64 robots. The experiment time was set at 700, with the world including 0 predators, 30 small objects, 30 large objects, 12 mobile obstacles, and 24 mobile obstacles. The total possible performance was 120 units. The graphs for these experiments are shown in Figures 7 and 8.

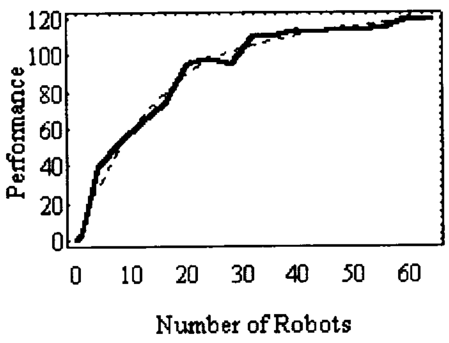


Figure 7: Team performance vs. team size.

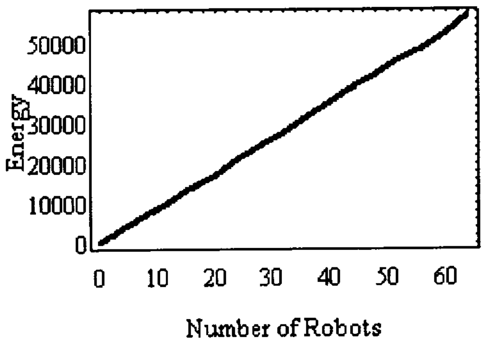


Figure 8: Team energy consumption vs. team size.

In these experiments the performance increases, as more robots are included in the team. The performance eventually levels off since the number of robots reaches a point where the maximum performance in the world is reached. The fitted curve for the performance is a degree four polynomial. The energy increase in the cases of larger teams is linear, since the energy consumptions of all robots are equal. Therefore the size of a robot team can be increased, up to a point where the performance levels off. The faster growth rate of the performance versus the energy consumption justifies the larger team size.

6. Conclusions

A new type of architecture for the control of autonomous mobile robots was presented in this paper. The Tropism-Based Cognitive Architecture is a simple and powerful method for enabling the robots to produce and perform actions based on their sensory input. A team of robots, equipped with this type of architecture was used in a number of realistic simulation experiments. These robots were able to perform a number of tasks, while surviving in a world that contained hostile, mobile predators. The robots located, processed, and collected objects, while navigating around stationary and mobile obstacle in an unstructured world. The work in progress includes a number of extensions to the architecture, and implementing and testing of the concepts on a group of real robots in the physical world.

Bibliography

[1] Agah, A. (1993). *Principles of Sociorobotics*. Technical Report IRIS-93-317, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California.

[2] Albus, J. S. (1991). Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21: 473-509.

[3] Arbib, M. A. (1989). *The Metaphorical Brain 2: Neural Networks and Beyond*. Wiley-Interscience, New York.

[4] Arkin, R. C. (1992). Cooperation without communication: multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9: 351-364.

[5] Beer, R. D. (1990). *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*. Academic Press, San Diego.

[6] Bekey, G. A. and Tomovic, R. (1986). Robot control by reflex actions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 240-246.

[7] Beni, G. and Hackwood, S. (1990). The maximum entropy principle and sensing in swarm intelligence. In Varela, F. J. and Bourgine, P. (Eds.) *Toward a Practice of Autonomous Systems*. MIT Press, Cambridge, Massachusetts, 153-160.

[8] Braitenberg, V. (1984). *Vehicles, Experiments in Synthetic Psychology*. MIT Press, Cambridge, Massachusetts.

[9] Brooks, R. A. (1989). A robot that walks: Emergent behaviors from a carefully evolved network. *Neural Computation*, 1: 253-262.

[10] Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chretien, L. (1991). The dynamics of collective sorting robot-like ants and ant-like robots. In Meyer, J.-A. and Wilson, S. W. (Eds.) *From Animals to Animats*. MIT Press, Cambridge, Massachusetts, 356-363.

[11] Fukuda, T., Ueyama, T., and Arai, F. (1992). Control strategies for cellular robotic network. In Levis, A. H. and Stephanou, H. E. (Eds.) *Distributed Intelligence Systems*. Pergamon Press, Oxford, 177-182.

[12] Kaelbling, L. P. (1992). Foundations of Learning in autonomous agents. *Robotics and Autonomous Systems*, 8: 131-144.

[13] Kawauchi, Y., Inaba, M., and Fukuda, T. (1993). A principle of distributed decision Making of cellular robotic system(CEBOT). In *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, 833-838.

[14] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33: 1-64.

[15] Maes, P. and Brooks, R. A. (1991). Learning to coordinate Behaviors. In Iyengar, S. S. and Elfes, A. (Eds.) *Autonomous Mobile Robots: Control, Planning, and Architecture*, IEEE Computer Society Press, Los Alamitos, California, 224-230.

[16] Mataric, M. J. (1992). Minimizing complexity in controlling a mobile robot population. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 830-835.

[17] Walter, W. G. (1953). *The Living Brain*. W. W. Norton & Company, Inc., New York.

[18] Wilson, E. O. (1980). *Sociobiology: The Abridged Edition*. The Belknap Press, Cambridge, Massachusetts.

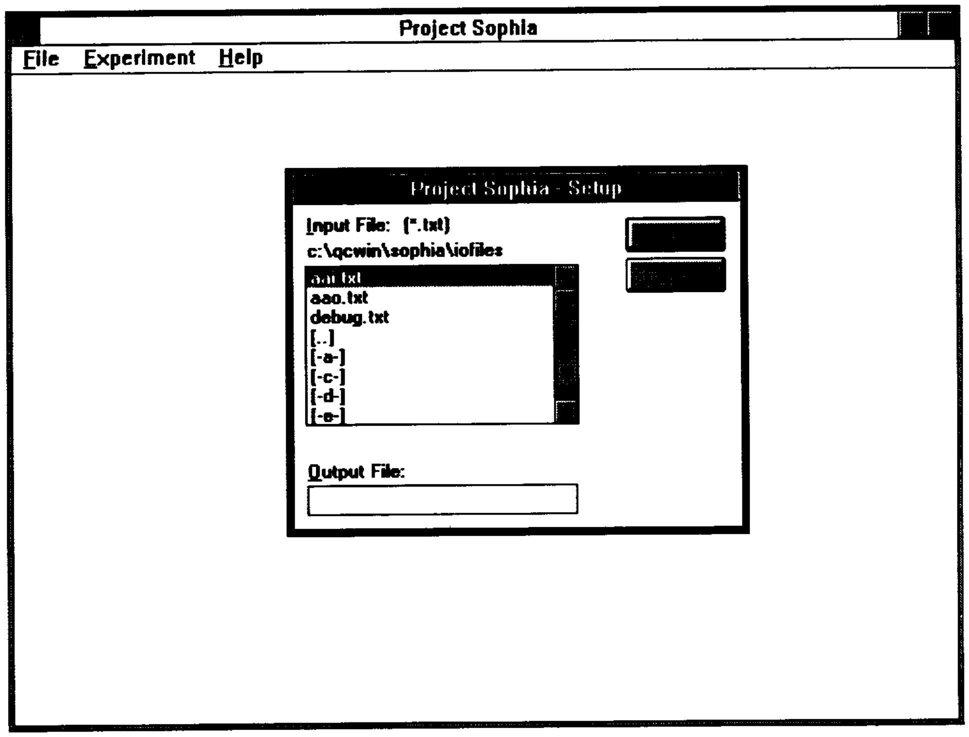


Figure 9: The setup screen for an experiment.

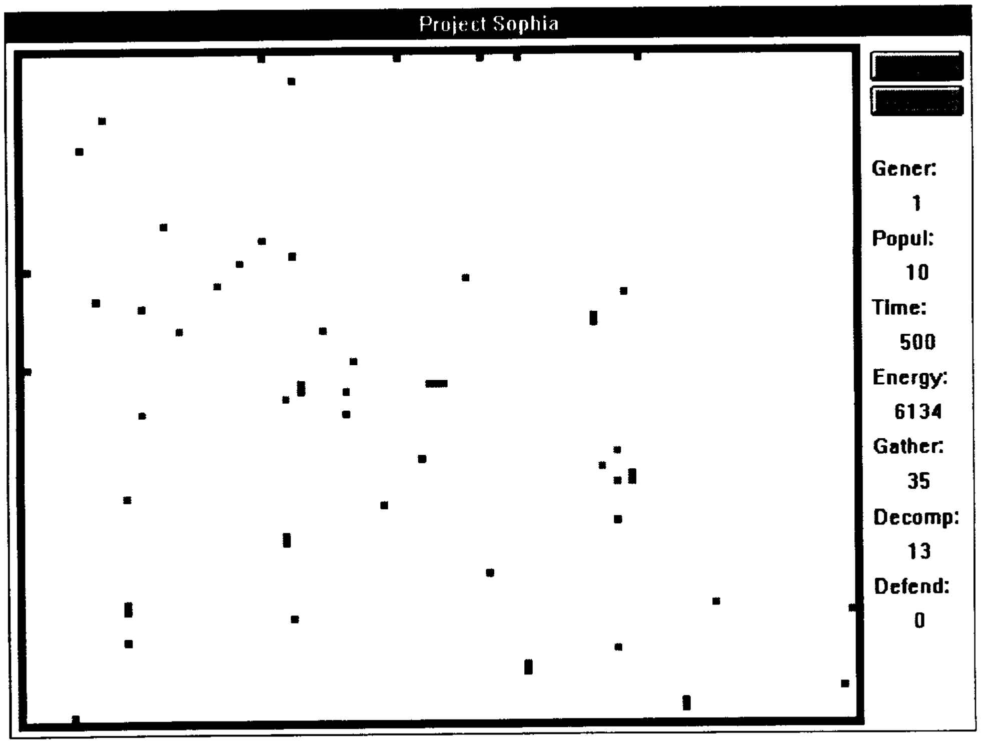


Figure 10: The experiment's world and its entities.

6054

N94- 30559

AIAA-94-1208-CP

BUILDING BRAINS FOR BODIES

Rodney Allen Brooks and **Lynn Andrea Stein**
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We describe a project to capitalize on newly available levels of computational resources in order to understand human cognition. We will build an integrated physical system including vision, sound input and output, and dextrous manipulation, all controlled by a continuously operating large scale parallel MIMD computer. The resulting system will learn to "think" by building on its bodily experiences to accomplish progressively more abstract tasks. Past experience suggests that in attempting to build such an integrated system we will have to fundamentally change the way artificial intelligence, cognitive science, linguistics, and philosophy think about the organization of intelligence. We expect to be able to better reconcile the theories that will be developed with current work in neuroscience.

Project Overview

We propose to build an integrated physical humanoid robot including active vision, sound input and output, dextrous manipulation, and the beginnings of language, all controlled by a continuously operating large scale parallel MIMD computer. This project will capitalize on newly available levels of computational resources in order to meet two goals: an engineering goal of building a prototype general purpose flexible and dextrous autonomous robot and a scientific goal of understanding human cognition. While there have been previous attempts at building kinematically humanoid robots, none have attempted the embodied construction of an autonomous intelligent robot; the requisite computational power simply has not previously been available.

The robot will be coupled into the physical world with high bandwidth sensing and fast servo-controlled actuators, allowing it to interact with the world on a human time scale. A shared time scale will open up new possibilities for how humans use robots as assistants, as well as allowing us to design the robot to learn new behaviors under human feedback such as human manual guidance and vocal approval. One of our engineering goals is to determine the architectural requirements sufficient for an enterprise of this type. Based on our earlier work on mobile robots, our expectation is that the constraints may be different than those that are often assumed for large scale parallel computers. If ratified, such a conclusion could have important impacts on the design of future sub-families of large machines.

Recent trends in artificial intelligence, cognitive science, neuroscience, psychology, linguistics, and sociology are converging on an anti-objectivist, body-based approach to abstract cognition. Where traditional approaches in these fields advocate an objectively specifiable reality—brain-in-a-box, independent of bodily constraints—these newer approaches insist that intelligence cannot be separated from the subjective experience of a body. The humanoid robot provides the necessary substrate for a serious exploration of the subjectivist—body-based—hypotheses.

There are numerous specific cognitive hypotheses that could be implemented in one or more of the humanoids that will be built during the five-year project. For example, we can vary the extent to which the robot is programmed with an attentional preference for some images or sounds, and the extent to which the robot is programmed to learn to selectively attend to environmental input as a by-product of goal attainment (e.g., successful manipulation of objects) or reward by humans. We can compare the behavioral result of constructing a humanoid around different hypotheses of cortical representation, such as *coincidence detection* versus *interpolating memory* versus *sequence seeking in counter streams* versus *time-locked multi-regional retroactivation*. In the later years of the project we can connect with theories of consciousness by demonstrating that humanoids designed to continuously act on immediate sensory data (as suggested by Dennett's *multiple drafts* model) show more human-like behavior than robots designed to construct an elaborate world model.

The act of building and programming behavior-based robots will force us to face not only issues of interfaces between traditionally assumed modularities, but even the idea of modularity itself. By reaching across traditional boundaries and tying together many sensing and acting modalities, we will quickly illuminate shortcomings in the standard models, shedding light on formerly unrealized sociologically shared, but incorrect, assumptions.

Background: the power of enabling technology

An enabling technology—such as the brain that we will build—has the ability to revolutionize science. A recent example of the far-reaching effects of such technological advances is the field of mobile robotics. Just as the advent of cheap and accessible mobile robotics dramatically altered our conceptions of intelligence in the last decade, we believe that current high-performance computing technology makes the present an opportune time for the construction of a similarly significant integrated intelligent system.

Over the last eight years there has been a renewed interest in building experimental mobile robot systems that operate in unadorned and unmodified natural and unstructured environments. The enabling technology for this was the single chip micro-computer. This made it possible for relatively small groups to build serviceable robots largely with graduate student power, rather than the legion of engineers that had characterized earlier efforts along these lines in the late sixties. The accessibility of this technology inspired academic researchers to take seriously the idea of building systems that would work in the real world.

The act of building and programming behavior-based robots fundamentally changed our understanding of what is difficult and what is easy. The effects of this work on traditional artificial intelligence can be seen in innumerable areas. Planning research has undergone a major shift from static planning to deal with "reactive planning." The emphasis in computer vision has moved from recovery from single images or canned sequences of images to active—or animate—vision, where the observer is a participant in the world controlling the imaging process in order to simplify the processing requirements. Generally, the focus within AI has shifted from centralized systems to distributed systems. Further, the work on behavior-based mobile robots has also had a substantial effect on many other fields (e.g., on the design of planetary science missions, on silicon micro-machining, on artificial life, and on cognitive science). There has also been considerable interest from neuroscience circles, and we are just now starting to see some bi-directional feedback there.

The grand challenge that we wish to take up is to make the quantum leap from experimenting with mobile robot systems to an almost humanoid integrated head system with saccading foveated vision, facilities for sound processing and sound production, and a compliant, dextrous manipulator. The enabling technology is massively parallel computing; our brain will have large numbers of processors dedicated to particular sub-functions, and interconnected by a fixed topology network.

Scientific Questions

Building an android, an autonomous robot with humanoid form, has been a recurring theme in science fiction from the inception of the genre with Frankenstein, through the moral dilemmas infesting positronic brains, the human but not really human C3PO and the ever present desire for real humanness as exemplified by Commander Data. Their bodies have ranged from that of a recycled actual human body through various degrees of mechanical sophistication to ones that are indistinguishable (in the stories) from real ones. And perhaps the most human of all the imagined robots, HAL-9000, did not even have a body.

While various engineering enterprises have modeled their artifacts after humans to one degree or another (e.g., WABOT-II at Waseda University and the space station tele-robotic servicer of Martin-Marietta) no one has seriously tried to couple human like cognitive processes to these systems. There has been an implicit, and sometimes explicit, assumption, even from the days of Turing (see Turing (1970)*) that the ultimate goal of artificial intelligence research was to build an android. There have been many studies relating brain models to computers (Berkeley 1949), cybernetics (Ashby 1956), and artificial intelligence (Arbib 1964), and along the way there have always been semi-popular scientific books discussing the possibilities of actually building real 'live' androids (Caudill (1992) is perhaps the most recent).

This proposal concerns a plan to build a series of robots that are both humanoid in form, humanoid in function, and to some extent humanoid in computational organization. While one cannot deny the romance of such an enterprise we are realistic enough to know that we can but scratch the surface of just a few of the scientific and technological problems involved in building the ultimate humanoid given the time scale and scope of our proposal, and given the current state of our knowledge.

The reason that we should try to do this at all is that for the first time there is plausibly enough computation available. High performance parallel computation gives us a new tool that those before us have not had available and that our contemporaries have chosen not to use in such a grand attempt. Our previous experience in attempting to emulate much simpler organisms than humans suggests that in attempting to build such systems we will have to fundamentally change the way artificial intelligence, cognitive science, psychology, and linguistics think about the organiza-

*Different sources cite 1947 and 1948 as the time of writing, but it was not published until long after his death.

tion of intelligence. As a result, some new theories will have to be developed. We expect to be better able to reconcile the new theories with current work in neuroscience. The primary benefits from this work will be in the striving, rather than in the constructed artifact.

Brains

Our goal is to take advantage of the new availability of massively parallel computation in dedicated machines. We need parallelism because of the vast amounts of processing that must be done in order to make sense of a continuous and rich stream of perceptual data. We need parallelism to coordinate the many actuation systems that need to work in synchrony (e.g., the ocular system and the neck must move in a coordinated fashion at time to maintain image stability) and which need to be servoed at high rates. We need parallelism in order to have a continuously operating system that can be upgraded without having to recompile, reload, and restart all of the software that runs the stable lower level aspects of the humanoid. And finally we need parallelism for the cognitive aspects of the system as we are attempting to build a "brain" with more capability than can fit on any existing single processor.

But in real-time embedded systems there is yet another necessary reason for parallelism. It is the fact that there are many things to be attended to, happening in the world continuously, independent of the agent. From this comes the notion of an agent being situated in the world. Not only must the agent devote attention to perhaps hundreds of different sensors many times per second, but it must also devote attention "down stream" in the processing chain in many different places at many times per second as the processed sensor data flows through the system. The actual amounts of computation needed to be done by each of these individual processes is in fact quite small, so small that originally we formalized them as augmented finite state machines (Brooks 1986), although more recently we have thought of them as real-time rules (Brooks 1990*a*). They are too small to have a complete processor devoted to them in any machine beyond a CM-2, and even there the processors would be mostly idle. A better approach is to simulate parallelism in a single conventional processor with its own local memory.

For instance, Ferrell (1993) built a software system to control a 19 actuator six legged robot using about 60 of its sensors. She implemented it as more than 1500 parallel processes running on a single Phillips 68070. (It communicated with 7 peripheral processors which handled sensor data collection and 100Hz motor servoing.) Most of these parallel processes ran at rates varying between 10 and 25 Hertz. Each time each process ran, it took at most a few dozen instructions before blocking, waiting either for the passage of time or for some other process to send it a message. Clearly, low cost context switching was important.

The underlying computational model used on that robot—and with many tens of other autonomous mobile robots we have built—consisted of networks of message-passing augmented finite state machines. Each of these AFSMs was a separate process. The messages were sent over predefined 'wires' from a specific transmitting to a specific receiving AFSM. The messages were simple numbers (typically 8 bits) whose meaning depended on the designs of both the transmitter and the receiver. An AFSM had additional registers which held the most recent incoming message on any particular wire. This gives a very simple model of parallelism, even simpler than that of CSP (Hoare 1985). The registers could have their values fed into a local combinatorial circuit to produce new values for registers or to provide an output message. The network of AFSMs was totally asynchronous, but individual AFSMs could have fixed duration monostables which provided for dealing with the flow of time in the outside world. The behavioral competence of the system was improved by adding more behavior-specific network to the existing network. This process was called *layering*. This was a simplistic and crude analogy to evolutionary development. As with evolution, at every stage of the development the systems were tested. Each of the layers was a behavior-producing piece of network in its own right, although it might implicitly rely on the presence of earlier pieces of network. For instance, an *explore* layer did not need to explicitly avoid obstacles, as the designer knew that a previous *avoid* layer would take care of it. A fixed priority arbitration scheme was used to handle conflicts.

On top of the AFSM substrate we used another abstraction known as the Behavior Language, or BL (Brooks 1990*a*), which was much easier for the user to program with. The output of the BL compiler was a standard set of augmented finite state machines; by maintaining this compatibility all existing software could be retained. When programming in BL the user has complete access to full Common Lisp as a metalanguage by way of a macro mechanism. Thus the user could easily develop abstractions on top of BL, while still writing programs which compiled down to networks of AFSMs. In a sense, AFSMs played the role of assembly language in normal high level computer languages. But the structure of the AFSM networks enforced a programming style which naturally compiled into very efficient small processes. The structure of the Behavior Language enforced a modularity where data sharing was restricted to smallish sets of AFSMs, and whose only interfaces were essentially asynchronous 1-deep buffers.

In the humanoid project we believe much of the computation, especially for the lower levels of the system, will naturally be of a similar nature. We expect to perform different experiments where in some cases the higher level computations are of the same nature and in other cases the higher levels will be much more symbolic in nature, although the symbolic bindings will be restricted to within individual processors. We need to use software and hardware environments which give support to these requirements without sacrificing the high levels of performance of which we wish to make use.

Software

For the software environment we have a number of requirements:

- There should be a good software development environment.
- The system should be completely portable over many hardware environments, so that we can upgrade to new parallel machines over the lifetime of this project.
- The system should provide efficient code for perceptual processing such as vision.
- The system should let us write high level symbolic programs when desired.
- The system language should be a standardized language that is widely known and understood.

In summary our software environment should let us gain easy access to high performance parallel computation.

We have chosen to use Common Lisp (Steele Jr. 1990) as the substrate for all software development. This gives us good programming environments including type checked debugging, rapid prototyping, symbolic computation, easy ways of writing embedded language abstractions, and automatic storage management. We believe that Common Lisp is superior to C (the other major contender) in all of these aspects.

The problem then is how to use Lisp in a massively parallel machine where each node may not have the vast amounts of memory that we have become accustomed to feeding Common Lisp implementations on standard Unix boxes.

We have a long history of building high performance Lisp compilers (Brooks, Gabriel & Steele Jr. 1982), including one of the two most common commercial Lisp compilers on the market; Lucid Lisp—Brooks, Posner, McDonald, White, Benson & Gabriel (1986).

Recently we have developed L (Brooks 1993), a retargetable small efficient Lisp which is a downwardly compatible subset of Common Lisp. When compiled for a 68000 based machine the load image (without the compiler) is only 140K bytes, but includes multiple values, strings, characters, arrays, a simplified but compatible package system, all the "ordinary" aspects of **format**, backquote and comma, **setf** etc., full Common Lisp lambda lists including optionals and keyword arguments, macros, an inspector, a debugger, **defstruct** (integrated with the inspector), **block**, **catch**, and **throw**, etc., full dynamic closures, a full lexical interpreter, floating point, fast garbage collection, and so on. The compiler runs in time linear in the size of an input expression, except in the presence of lexical closures. It nevertheless produces highly optimized code in most cases. L is missing **flet** and **labels**, generic arithmetic, bignums, rationals, complex numbers, the library of sequence functions (which can be written within L) and esoteric parts of **format** and packages.

The L system is an intellectual descendent of the dynamically retargetable Lucid Lisp compiler (Brooks et al. 1986) and the dynamically retargetable Behavior Language compiler (Brooks 1990*a*). The system is totally written in L with machine dependent backends for retargetting. The first backend is for the Motorola 68020 (and upwards) family, but it is easily retargeted to new architectures. The process consists of writing a simple machine description, providing code templates for about 100 primitive procedures (e.g., fixed precision integer **+**, *****, **=**, etc., string indexing **CHAR** and other accessors, **CAR**, **CDR**, etc.), code macro expansion for about 20 pseudo instructions (e.g, procedure call, procedure exit, checking correct number of arguments, linking **CATCH** frames, etc.) and two corresponding sets of assembler routines which are too big to be expanded as code templates every time, but are so critical in speed that they need to be written in machine language, without the overhead of a procedure call, rather than in Lisp (e.g., **CONS**, spreading of multiple values on the stack, etc.). There is a version of the I/O system which operates by calling C routines (e.g., **fgetchar**, etc.; this is how the Macintosh version of L runs) so it is rather simple to port the system to any hardware platform we might choose to use in the future.

Note carefully the intention here: L is to be the delivery vehicle running on the brain hardware of the humanoid, potentially on hundreds or thousands of small processors. Since it is fully downward compatible with Common Lisp however, we can carry out code development and debugging on standard work stations with full programming environments (e.g., in Macintosh Common Lisp, or Lucid Common Lisp with Emacs 19 on a Unix box, or in the Harlequin programming environment on a Unix box). We can then dynamically link code into the running system on our parallel processors.

There are two remaining problems: (1) how to maintain super critical real-time performance when using a Lisp system without hard ephemeral garbage collection, and (2) how to get the level of within-processor parallelism described earlier.

The structure of L's implementation is such that multiple independent heaps can be maintained within a single address space, sharing all the code and data segments of the Lisp proper. In this way super-critical portions of a system can be placed in a heap where no consing is occurring, and hence there is no possibility that they will be blocked by garbage collection.

The Behavior Language (Brooks 1990*a*) is an example of a compiler which builds special purpose static schedulers for low overhead parallelism. Each process ran until blocked and the syntax of the language forced there to always be a blocking condition, so there was no need for pre-emptive scheduling. Additionally the syntax and semantics of the language guaranteed that there would be zero stack context needed to be saved when a blocking condition was reached. We will need to build a new scheduling system with L to address similar issues in this project. To fit in with the philosophy of the rest of the system it must be a dynamic scheduler so that new processes can be added and deleted as a user types to the Lisp listener of a particular processor. Reasonably straightforward data structures can keep these costs to manageable levels. It is rather straightforward to build a phase into the L compiler which can recognize the situations described above. Thus it is straightforward to implement a set of macros which will provide a language abstraction on top of Lisp which will provide all the functionality of the Behavior Language and which will additionally let us have dynamic scheduling. Almost certainly a pre-emptive scheduler will be needed in addition, as it would be difficult to enforce a computation time limit syntactically when Common Lisp will essentially be available to the programmer—at the very least the case of the pre-emptive scheduler having to strike down a process will be useful as a safety device, and will also act as a debugging tool for the user to identify time critical computations which are stressing the bounded computation style of writing. In other cases static analysis will be able to determine maximum stack requirements for a particular process, and so heap allocated stacks will be usable.[†]

The software system so far described will be used to implement crude forms of 'brain models', where computations will be organized in ways inspired by the sorts of anatomical divisions we see occurring in animal brains. Note that we are not saying we will build a model of a particular brain, but rather there will be a modularity inspired by such components as visual cortex, auditory cortex, etc., and within and across those components there will be further modularity, e.g., a particular subsystem to implement the vestibulo-ocular response (VOR).

Thus besides on-processor parallelism we will need to provide a modularity tool that packages processes into groups and limits data sharing between them. Each package will reside on a single processor, but often processors will host many such packages. A package that communicates with another package should be insulated at the syntax level from knowing whether the other package is on the same or a different processor. The communication medium between such packages will again be 1-deep buffers without queuing or receipt acknowledgment—any such acknowledgment will need to be implemented as a backward channel, much as we see throughout the cortex (Churchland & Sejnowski 1992). This packaging system can be implemented in Common Lisp as a macro package.

We expect all such system level software development to be completed in the first twelve months of the project.

Computational Hardware

The computational model presented in the previous section is somewhat different from that usually assumed in high performance parallel computer applications. Typically (Cypher, Ho, Konstantinidou & Messina 1993) there is a strong bias on system requirements from the sort of benchmarks that are used to evaluate performance. The standard benchmarks for modern high performance computation seem to be Fortran codes for hydrodynamics, molecular simulations, or graphics rendering. We are proposing a very different application with very different requirements; in particular we require real-time response to a wide variety of external and internal events, we require good symbolic computation performance, we require only integer rather than high performance floating point operations,[‡] we require delivery of messages only to specific sites determined at program design time, rather than at run-time, and we require the ability to do very fast context switches because of the large number of parallel processes that we intend to run on each individual processor.

The fact that we will not need to support pointer references across the computational substrate will mean that we can rely on much simpler, and therefore higher performance, parallel computers than many other researchers—we will not have to worry about a consistent global memory, cache coherence, or arbitrary message routing. Since these are different requirements than those that are normally considered, we have to make some measurements with actual programs before we can we can make an intelligent off the shelf choice of computer hardware.

In order to answer some of these questions we are currently building a zero-th generation parallel computer. It is being built on a very low budget with off the shelf components wherever possible (a few fairly simple printed circuit boards need to be fabricated). The processors are 16Mhz Motorola 68332s on a standard board built by Vesta Technology. These plug 16 to a backplane. The backplane provides each processor with six communications ports (using the integrated timing processor unit to generate the required signals

[†]The problem with heap allocated stacks in the general case is that there will be no overflow protection into the rest of heap.

[‡]Consider the dynamic range possible in single signal channels in the human brain and it soon becomes apparent that all that we wish to do is certainly achievable with neither span of 600 orders of magnitude, or 47 significant binary digits.

along with special chip select and standard address and data lines) and a peripheral processor port. The communications ports will be hand-wired with patch cables, building a fixed topology network. (The cables incorporate a single dual ported RAM (8K by 16 bits) that itself includes hardware semaphores writable and readable by the two processors being connected.) Background processes running on the 68332 operating system provide sustained rate transfers of 60Hz packets of 4K bytes on each port, with higher peak rates if desired. These sustained rates do consume processing cycles from the 68332. On non-vision processors we expect much lower rates will be needed, and even on vision processors we can probably reduce the packet frequency to around 15Hz. Each processor has an operating system, L, and the dynamic scheduler residing in 1M of EPROM. There is 1M of RAM for program, stack and heap space. Up to 256 processors can be connected together.

Up to 16 backplanes can be connected to a single front end processor (FEP) via a shared 500K baud serial line to a SCSI emulator. A large network of 68332s can span many FEPs if we choose to extend the construction of this zero-th prototype. Initially we will use a Macintosh as a FEP. Software written in Macintosh Common Lisp on the FEP will provide disk I/O services to the 68332's, monitor status and health packets from them, and provide the user with a Lisp listener to any processor they might choose.

The zero-th version uses the standard Motorola SPI (serial peripheral interface) to communicate with up to 16 Motorola 6811 processors per 68332. These are a single chip processor with onboard EEPROM (2K bytes) and RAM (256 bytes), including a timer system, an SPI interface, and 8 channels of analog to digital conversion. We are building a small custom board for this processor that includes opto-isolated motor drivers and some standard analog support for sensors[§].

We expect our first backplane to be operational by August 1st, 1993 so that we can commence experiments with our first prototype body. We will collect statistics on inter-processor communication throughput, effects of latency, and other measures so that we can better choose a larger scale parallel processor for more serious versions of the humanoid.

In the meantime, however, there are certain developments on the horizon within the MIT Artificial Intelligence Lab which we expect to capitalize upon in order to dramatically upgrade our computational systems for early vision, and hence the resolution at which we can afford to process images in real time. The first of these, expected in the fall will be a somewhat similar distributed processing system based on the much higher performance Texas Instrument C40, which comes with built in support for fixed topology message passing. We expect these systems to be available in the Fall '93 timeframe. In October '94 we expect to be able to make use of the Abacus system, a bit level reconfigurable vision front-end processor being built under ARPA sponsorship which promises Tera-op performance on 16 bit fixed precision operands. Both these systems will be simply integrable with our zero-th order parallel processor via the standard dual-ported RAM protocol that we are using.

[§]We currently have 28 operational robots in our labs each with between 3 and 5 of these 6811 processors, and several dozen other robots with at least 1 such processor on board. We have great experience in writing compiler backends for these processors (including BL) and great experience in using them for all sorts of servoing, sensor monitoring, and communications tasks.

Bodies

As with the computational hardware, we are also currently engaged in building a zero-th generation body for early experimentation and design refinement towards more serious constructions within the scope of this proposal. We are presently limited by budgetary constraints to building an immobile, armless, deaf, torso with only black and white vision.

In the following subsections we outline the constraints and requirements on a full scale humanoid body and also include where relevant details of our zero-th level prototype.

Eyes

There has been quite a lot of recent work on *animate vision* using saccading stereo cameras, most notably at Rochester (Ballard 1989), (Coombs 1992), but also more recently at many other institutions, such as Oxford University.

The humanoid needs a head with high mechanical performance eyeballs and foveated vision if it is to be able to participate in the world with people in a natural way. Even our earliest heads will include two eyes, with foveated vision, able to pan and tilt as a unit, and with independent saccading ability (three saccades per second) and vergence control of the eyes. Fundamental vision based behaviors will include a visually calibrated vestibular-ocular reflex, smooth pursuit, visually calibrated saccades, and object centered foveal relative depth stereo. Independent visual systems will provide peripheral and foveal motion cues, color discrimination, human face pop-outs, and eventually face recognition. Over the course of the project, object recognition based based on "representations" from body schemas and manipulation interactions will be developed. This is completely different from any conventional object recognition schemes, and can not be attempted without an integrated vision and manipulation environment as we propose.

The eyeballs need to be able to saccade up to about three times per second, stabilizing for 250ms at each stop. Additionally the yaw axes should be controllable for vergence to a common point and drivable in a manner appropriate for smooth pursuit and for image stabilization as part of a vestibulo-ocular response (VOR) to head movement. The eyeballs do not need to be force or torque controlled but they do need good fast position and velocity control. We have previously built a single eyeball, *A-eye*, on which we implemented a model of VOR, ocular-kinetic response (OKR) and saccades, all of which used dynamic visually based calibration (Viola 1990).

Other active vision systems have had both eyeballs mounted on a single tilt axis. We will begin experiments with separate tilt axes but if we find that relative tilt motion is not very useful we will back off from this requirement in later versions of the head.

The cameras need to cover a wide field of view, preferably close to 180 degrees, while also giving a foveated central region. Ideally the images should be RGB (rather than the very poor color signal of standard NTSC). A resolution of 512 by 512 at both the coarse and fine scale is desirable.

Our zero-th version of the cameras are black and white only. Each eyeball consists of two small lightweight cameras mounted with parallel axes. One gives a 115 degree field of view and the other gives a 20 degree foveated region. In order to handle the images in real time in our zero-th parallel processor we will subsample the images to be much smaller than the ideal.

Later versions of the head will have full RGB color cameras, wider angles for the peripheral vision, much finer grain sampling of the images, and perhaps a colinear optics set up using optical fiber cables and beam splitters. With more sophisticated high speed processing available we will also be able to do experiments with log-polar image representations.

Ears, Voice

Almost no work has been done on sound understanding, as distinct from speech understanding. This project will start on sound understanding to provide a much more solid processing base for later work on speech input. Early behavior layers will spatially correlate noises with visual events, and spatial registration will be continuously self calibrating. Efforts will concentrate on using this physical cross-correlation as a basis for reliably pulling out interesting events from background noise, and mimicking the cocktail party effect of being able to focus attention on particular sound sources. Visual correlation with face pop-outs, etc., will then be used to be able to extract human sound streams. Work will proceed on using these sounds streams to mimic infant's abilities to ignore language dependent irrelevances. By the time we get to elementary speech we will therefore have a system able to work in noisy environments and accustomed to multiple speakers with varying accents.

Sound perception will consist of three high quality microphones. (Although the human head uses only two auditory inputs, it relies heavily on the shape of the external ear in determining the vertical component of directional sound source.) Sound generation will be accomplished using a speaker.

Sound is critical for several aspects of the robot's activity. First, sound provides immediate feedback for motor manipulation and positioning. Babies learn to find and use their hands by batting at and manipulating toys that jingle and rattle. Adults use such cues as contact noises—the sound of an object hitting the table—to provide feedback to motor systems. Second, sound aids in socialization even before the emergence of language. Patterns such as turn-taking and mimicry are critical parts of children's development, and adults use guttural gestures to express attitudes and other conversational cues. Certain signal tones indicate encouragement or disapproval to all ages and stages of development. Finally, even pre-verbal children use sound effectively to convey intent; until our robots develop true language, other sounds will necessarily be a major source of communication.

Torsos

In order for the humanoid to be able to participate in the same sorts of body metaphors as are used by humans, it needs to have a symmetric human-like torso. It needs to be able to experience imbalance, feel symmetry, learn to coordinate head and body motion for stable vision, and be able to experience relief when it relaxes its body. Additionally the torso must be able to support the head, the arms, and any objects they grasp.

The torsos we build will initially have a three degree of freedom hip, with the axes passing through a common point, capable of leaning and twisting to any position in about three seconds—somewhat slower than a human. The neck will also have three degrees of freedom, with the axes passing through a common point which will also lie along the spinal axis of the body. The head will be capable of yawing at 90 degrees per second—less than peak human speed, but well within the range of natural human motions. As we build later versions we expect to increase these performance figures to more closely match the abilities of a human.

Apart from the normal sorts of kinematic sensors, the torso needs a number of additional sensors specifically aimed at providing input fodder for the development of bodily metaphors. In particular, strain gauges on the spine can give the system a feel for its posture and the symmetry of a particular configuration, plus a little information about any additional load the torso might bear when an arm picks up something heavy. Heat sensors on the motors and the motor drivers will give feedback as to how much work has been done by the body recently, and current sensors on the motors will give an indication of how hard the system is working instantaneously.

Our zero-th level torso is roughly 18 inches from the base of the spine to the base of the neck. This corresponds to a smallish adult. It uses DC motors with built in gearboxes. The main concern we have is how quiet it will be, as we do not want the sound perception system to be overwhelmed by body noise.

Later versions of the torsos will have touch sensors integrated around the body, will have more compliant motion, will be quieter, and will need to provide better cabling ducts so that the cables can all feed out through a lower body outlet.

Arms

The eventual manipulator system will be a compliant multi-degree of freedom arm with a rather simple hand. (A better hand would be nice, but hand research is not yet at a point where we can get an interesting, easy-to use, off-the-shelf hand.) The arm will be safe enough that humans can interact with it, handing it things and taking things from it. The arm will be compliant enough that the system will be able to explore its own body—for instance, by touching its head system—so that it will be able to develop its own body metaphors. The full design of the even the first pair of arms is not yet completely worked out, and current funding does not permit the inclusion of arms on the zero-th level humanoid. In this section, we describe our desiderata for the arms and hands.

We want the arms to be very compliant yet still able to lift weights of a few pounds so that they can interact with human artifacts in interesting ways. Additionally we want the arms to have redundant degrees of freedom (rather than the six seen in a standard commercial robot arm), so that in many circumstances we can 'burn' some of those degrees of freedom in order to align a single joint so that the joint coordinates and task coordinates very nearly match. This will greatly simplify control of manipulation. It is the sort of thing people do all the time: for example, when bracing an elbow or the base of the palm (or even their middle and last two fingers) on a table to stabilize the hand during some delicate (or not so delicate) manipulation.

The hands in the first instances will be quite simple; devices that can grasp from above relying heavily on mechanical compliance—they may have as few as one degree of control freedom.

More sophisticated, however, will be the sensing on the arms and hands. We will use forms of conductive rubber to get a sense of touch over the surface of the arm, so that it can detect (compliant) collisions it might participate in. As with the torso there will be liberal use of strain gauges, heat sensors and current sensors so that the system can have a 'feel' for how its arms are being used and how they are performing.

We also expect to move towards a more sophisticated type of hand in later years of this project. Initially, unfortunately, we will be forced to use motions of the upper joints of the arm for fine manipulation tasks. More sophisticated hands will allow us to use finger motions, with much lower inertias, to carry out these tasks.

Development Plan

We plan on modeling the brain at a level above the neural level, but below what would normally be thought of as the cognitive level.

We understand abstraction well enough to know how to engineer a system that has similar properties and connections to the human brain without having to model its detailed local wiring. At the same time it is clear from the literature that there is no agreement on how things are really organized computationally at higher or modular levels, or indeed whether it even makes sense to talk about modules of the brain (e.g., short term memory, and long term memory) as generative structures.

Nevertheless, we expect to be guided, or one might say inspired, by what is known about the high level connectivity within the human brain (although admittedly much of our knowledge actually comes from macaques and other primates and is only extrapolated to be true of humans, a problem of concern to some brain scientists (Crick & Jones 1993)). Thus for instance we expect to have identifiable clusters of processors which we will be able to point to and say they are performing a role similar to that of the cerebellum (e.g., refining gross motor commands into coordinated smooth motions), or the cortex (e.g., some aspects of searching generalization/specialization hierarchies in object recognition (Ullman 1991)).

At another level we will directly model human systems where they are known in some detail. For instance there is quite a lot known about the control of eye movements in humans (again mostly extrapolated from work with monkeys) and we will build in a vestibulo-ocular response (VOR), OKR, smooth pursuit, and saccades using the best evidence available on how this is organized in humans (Lisberger 1988).

A third level of modeling or inspiration that we will use is at the developmental level. For instance once we have some sound understanding developed, we will use models of what happens in child language development to explore ways of connecting physical actions in the world to a ground of language and the development of symbols (Bates 1979), (Bates, Bretherton & Snyder 1988), including indexical (Lempert & Kinsbourne 1985) and turn-taking behavior, interpretation of tone and facial expressions and the early use of memorized phrases.

Since we will have a number of faculty, post-doctoral fellows, and graduate students working on concurrent research projects, and since we will have a number of concurrently active humanoid robots, not all pieces that are developed will be intended to fit together exactly. Some will be incompatible experiments in alternate ways of building subsystems, or putting them together. Some will be pushing on particular issues in

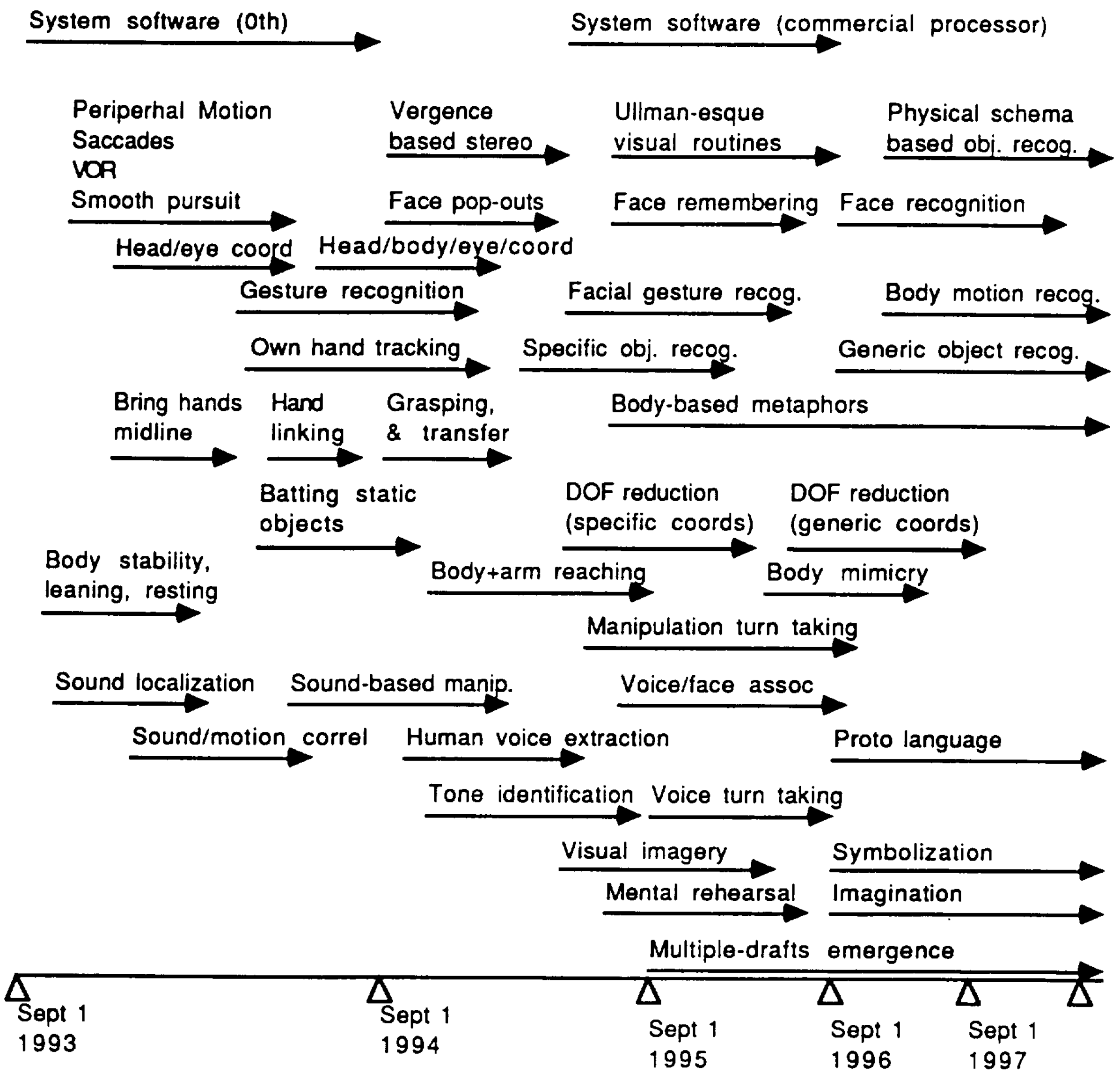

System software (0th)
System software (commercial processor)
Periperhal Motion
Saccades
VOR
Smooth pursuit
Vergence based stereo
Ullman-esque visual routines
Physical schema based obj. recog.
Face pop-outs
Face remembering
Face recognition
Head/eye coord
Head/body/eye/coord
Gesture recognition
Facial gesture recog.
Body motion recog.
Own hand tracking
Specific obj. recog.
Generic object recog.
Bring hands midline
Hand linking
Grasping, & transfer
Body-based metaphors
Batting static objects
DOF reduction (specific coords)
DOF reduction (generic coords)
Body stability, leaning, resting
Body+arm reaching
Body mimicry
Manipulation turn taking
Sound localization
Sound-based manip.
Voice/face assoc
Sound/motion correl
Human voice extraction
Proto language
Tone identification
Voice turn taking
Visual imagery
Symbolization
Mental rehearsal
Imagination
Multiple-drafts emergence
Sept 1 1993
Sept 1 1994
Sept 1 1995
Sept 1 1996
Sept 1 1997

Figure 1

language, say, that may not be very related to some particular other issues, e.g., saccades. Also, quite clearly, at this stage we can not have a development plan fully worked out for five years, as many of the early results will change the way we think about the problems and what should be the next steps.

In figure 1, we summarize our current plans for developing software systems on board our series of humanoids. In many cases there will be earlier work off-board the robots, but to keep clutter down in the diagram we have omitted that work here.

Acknowledgements

This paper has benefitted from conversations with and comments by Catherine Harris, Dan Dennett, Marcel Kinsbourne. We are also indebted to the members of our research groups, individually and collectively, who have shared their thoughts and enthusiasm.

References

Agre, P. E. & Chapman, D. (1987), Pengi: An Implementation of a Theory of Activity, in 'Proceedings of the Sixth National Conference on Artificial Intelligence', Morgan Kaufmann, Seattle, Washington, pp. 196–201.

Allen, J., Hendler, J. & Tate, A., eds (1990), Readings in Planning, Morgan Kaufmann, Los Altos, California.

Angle, C. M. & Brooks, R. A. (1990), Small Planetary Rovers, in 'IEEE/RSJ International Workshop on Intelligent Robots and Systems', Ikabara, Japan, pp. 383–388.

Arbib, M. A. (1964), Brains, Machines and Mathematics, McGraw-Hill, New York, New York.

Ashby, W. R. (1956), An Introduction to Cybernetics, Chapman and Hall, London, United Kingdom.

Ballard, D. H. (1989), Reference Frames for Active Vision, in 'Proceedings of the International Joint Conference on Artificial Intelligence', Detroit, Michigan, pp. 1635–1641.

Bates, E. (1979), The Emergence of Symbols, Academic Press, New York, New York.

Bates, E., Bretherton, I. & Snyder, L. (1988), From First Words to Grammar, Cambridge University Press, Cambridge, United Kingdom.

Beer, R. D. (1990), Intelligence as Adaptive Behavior, Academic Press, Cambridge, Massachusetts.

Berkeley, E. C. (1949), Giant Brains or Machines that Think, John Wiley & Sons, New York, New York.

Bickhard, M. H. (n.d.), How to Build a Machine with Emergent Representational Content, Unpublished manuscript, University of Texas, Austin.

Brachman, R. J. & Levesque, H. J., eds (1985), Readings in Knowledge Representation, Morgan Kaufmann, Los Altos, California.

Braddick, O., Atkison, J., Hood, B., Harkness, W. & an Faraneh Vargha-Khadem, G. J. (1992), 'Possible blindsight in infants lacking one cerebral hemisphere', Nature **360**, 461–463.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', IEEE Journal of Robotics and Automation **RA-2**, 14–23.

Brooks, R. A. (1989), 'A Robot That Walks: Emergent Behavior from a Carefully Evolved Network', Neural Computation 1(2), 253–262.

Brooks, R. A. (1990a), The Behavior Language User's Guide, Memo 1227, Massachusetts Institute of TechnologyArtificial Intelligence Lab, Cambridge, Massachusetts.

Brooks, R. A. (1990b), Elephants Don't Play Chess, in P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', MIT Press, Cambridge, Massachusetts, pp. 3–15.

Brooks, R. A. (1991a), Intelligence Without Reason, in 'Proceedings of the 1991 International Joint Conference on Artificial Intelligence', pp. 569–595.

Brooks, R. A. (1991b), 'New Approaches to Robotics', Science **253**, 1227–1232.

Brooks, R. A. (1993), L: A Subset of Common Lisp, Technical report, Massachusetts Institute of TechnologyArtificial Intelligence Lab.

Brooks, R. A., Gabriel, R. P. & Steele Jr., G. L. (1982), An Optimizing Compiler for Lexically Scoped Lisp, in 'Proceedings of the 1982 Symposium on Compiler Construction. ACM SIGPLAN', Boston, Massachusetts, pp. 261–275. Publised as ACM SIGPLAN Notices 17, 6 (June 1982).

Brooks, R. A., Posner, D. B., McDonald, J. L., White, J. L., Benson, E. & Gabriel, R. P. (1986), Design of An Optimizing Dynamically Retargetable Compiler for Common Lisp, in 'Proceedings of the 1986 ACM Symposium on Lisp and Functional Programming', Cambridge, Massachusetts, pp. 67–85.

Caudill, M. (1992), In Our Own Image: Building An Artificial Person, Oxford University Press, New York, New York.

Churchland, P. S. & Sejnowski, T. J. (1992), The Computational Brain, MIT Press, Cambridge, Massachusetts.

Connell, J. H. (1987), Creature Building with the Subsumption Architecture, in 'Proceedings of the International Joint Conference on Artificial Intelligence', Milan, Italy, pp. 1124–1126.

Connell, J. H. (1990), Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot, Academic Press, Cambridge, Massachusetts. also MIT TR-1151.

Coombs, D. J. (1992), Real-time Gaze Holding in Binocular Robot Vision, PhD thesis, University of Rochester, Department of CS, Rochester, New York.

Crick, F. & Jones, E. (1993), 'Backwardness of human neuroanatomy', Nature **361**, 109–110.

Cypher, R., Ho, A., Konstantinidou, S. & Messina, P. (1993), Architectural Requirements of Parallel Scientific Applications with Explicit Communication, in 'IEEE Proceedings of the 20th International Symposium on Computer Architecture', San Diego, California, pp. 2–13.

Damasio, H. & Damasio, A. R. (1989), Lesion Analysis in Neuropsychology, Oxford University Press, New York, New York.

Dennett, D. C. (1991), Consciousness Explained, LIttle, Brown, Boston, Massachusetts.

Dennett, D. C. & Kinsbourne, M. (1992), 'Time and the Observer: The Where and When of Consciousness in the Brain', Brain and Behavioral Sciences **15**, 183–247.

Drescher, G. L. (1991), Made-Up Minds: A Constructivist Approach to Artificial Intelligence, MIT Press, Cambridge, Massachusetts.

Edelman, G. M. (1987), Neural Darwinsim: The Theory of Neuronal Group Selection, Basic Books, New York, New York.

Edelman, G. M. (1989), The Remembered Present: A Biological Theory of Consciousness, Basic Books, New York, New York.

Edelman, G. M. (1992), Bright Air, Brilliant Fire: On the Matter of Mind, Basic Books, New York, New York.

Fendrich, R., Wessinger, C. M. & Gazzaniga, M. S. (1992), 'Residual Vision in a Scotoma: Implications for Blindsight', Science **258**, 1489–1491.

Ferrell, C. (1993), Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators, Master's thesis, MIT, Department of EECS, Cambridge, Massachusetts.

Fodor, J. A. (1983), The Modularity of Mind, Bradford Books, MIT Press, Cambridge, Massachusetts.

Harris, C. L. (1991), Parallel Distributed Processing Models and Metaphors for Language and Development, PhD thesis, University of California, Department of Cognitive Science, San Diego, California.

Haugeland, J. (1985), Artificial Intelligence: The Very Idea, MIT Press, Cambridge, Massachusets.

Hoare, C. A. R. (1985), Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, New Jersey.

Hobbs, J. & Moore, R., eds (1985), Formal Theories of the Commonsense World, Ablex Publishing Co., Norwood, New Jersey.

Horswill, I. D. (1993), Specialization of Perceptual Processes, PhD thesis, MIT, Department of EECS, Cambridge, Massachusetts.

Horswill, I. D. & Brooks, R. A. (1988), Situated Vision in a Dynamic World: Chasing Objects, in 'Proceedings of the Seventh Annual Meeting of the American Association for Artificial Intelligence', St. Paul, Minnesota, pp. 796–800.

Johnson, M. (1987), The Body In The Mind, University of Chicago Press, Chicago, Illinois.

Kinsbourne, M. (1987), Mechanisms of unilateral neglect, in M. Jeannerod, ed., 'Neurophysiological and Neuropsychological Aspects of Spatial Neglect', Elsevier, North Holland.

Kinsbourne, M. (1988), Integrated field theory of consciousness, in A. Marcel & E. Bisiach, eds, 'The Concept of Consciousness in Contemporary Science', Oxford University Press, London, England.

Kosslyn, S. (1993), Image and brain: The resolution of the imagery debate, Harvard University Press, Cambridge, Massachusetts.

Kuipers, B. & Byun, Y.-T. (1991), 'A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations', Robotics and Autonomous Systems **8**, 47–63.

Lakoff, G. (1987), Women, Fire, and Dangerous Things, University of Chicago Press, Chicago, Illinois.

Lakoff, G. & Johnson, M. (1980), Metaphors We Live By, University of Chicago Press, Chicago, Illinois.

Langacker, R. W. (1987), Foundations of cognitive grammar, Volume 1, Stanford University Press, Palo Alto, California.

Lempert, H. & Kinsbourne, M. (1985), 'Possible origin of speech in selective orienting', Psychological Bulletin **97**, 62–73.

Lisberger, S. G. (1988), 'The neural basis for motor learning in the vestibulo-ocular reflex in monkeys', Trends in Neuroscience 11, 147–152.

Marr, D. (1982), Vision, W. H. Freeman, San Francisco, California.

Mataric, M. J. (1992a), Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, in 'Proceedings of the Second International Conference on Simulation of Adaptive Behavior', MIT Press, Cambridge, Massachusetts, pp. 432–441.

Mataric, M. J. (1992b), 'Integration of Representation Into Goal-Driven Behavior-Based Robots', IEEE Journal of Robotics and Automation **8**(3), 304–312.

McCarthy, R. A. & Warrington, E. K. (1988), 'Evidence for Modality-Specific Systems in the Brain', Nature **334**, 428–430.

McCarthy, R. A. & Warrington, E. K. (1990), Cognitive Neuropsychology, Academic Press, San Diego, California.

Minsky, M. (1986), The Society of Mind, Simon and Schuster, New York, New York.

Minsky, M. & Papert, S. (1969), Perceptrons, MIT Press, Cambridge, Massachusetts.

Newcombe, F. & Ratcliff, G. (1989), Disorders of Visupspatial Analysis, in 'Handbook of Neuropsychology, Volume 2', Elsevier, New York, New York.

Newell, A. & Simon, H. A. (1981), Computer Science as Empirical Inquiry: Symbols and Search, in J. Haugeland, ed., 'Mind Design', MIT Press, Cambridge, Massachusetts, chapter 1, pp. 35–66.

Penrose, R. (1989), The Emporer's New Mind, Oxford University Press, Oxford, United Kingdom.

Philip Teitelbaum, V. C. P. & Pellis, S. M. (1990), Can Allied Reflexes Promote the Integration of a Robot's Behavior, in 'Proceedings of the First International Conference on Simulation of Adaptive Behavior', MIT Press, Cambridge, Massachusetts, pp. 97–104.

Pomerleau, D. A. (1991), 'Efficient Training of Artificial Neural Networks for Autonomous Navigation', Neural Computation.

Rosenblatt, F. (1962), Principles of Neurodynamics, Spartan, New York, New York.

Rosenschein, S. J. & Kaelbling, L. P. (1986), The Synthesis of Digital Machines with Provable Epestemic Properties, in J. Y. Halpern, ed., 'Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge', Morgan Kaufmann, Monterey, California, pp. 83–98.

Rumelhart, D. E. & McClelland, J. L., eds (1986), Parallel Distributed Processing, MIT Press, Cambridge, Massachusetts.

Searle, J. R. (1992), The Rediscovery of the Mind, MIT Press, Cambridge, Massachusetts.

Simon, H. A. (1969), The Sciences of the Artificial, MIT Press, Cambridge, Massachusetts.

Springer, S. P. & Deutsch, G. (1981), Left Brain, Right Brain, W.H. Freeman and Company, New York.

Steele Jr., G. L. (1990), Common Lisp, The Language, second edn, Digital Press.

Stein, L. A. (to appear), 'Imagination and Situated Cognition', Journal of Experimental and Theoretical Artificial Intelligence.

Turing, A. M. (1970), Intelligent Machinery, in B. Meltzer & D. Michie, eds, 'Machine Intelligence 5', American Elsevier Publishing, New York, New York, pp. 3–23.

Ullman, S. (1991), Sequence-Seeking and Counter Streams: A Model for Information Processing in the Cortex, Memo 1311, Massachusetts Institute of TechnologyArtificial Intelligence Lab, Cambridge, Massachusetts.

Viola, P. A. (1990), Adaptive Gaze Control, Master's thesis, MIT, Department of EECS, Cambridge, Massachusetts.

Weiskrantz, L. (1986), Blindsight, Oxford University Press, Oxford, United Kingdom.

Yanco, H. & Stein, L. A. (1993), An Adaptive Communication Protocol for Cooperating Mobile Robots, in J.-A. Meyer, H. Roitblat & S. Wilson, eds, 'From Animals to Animats: Proceedings of the Second Conference on the Simulation of Adaptive Behavior', MIT Press, Cambridge, Massachusetts, pp. 478–485.

-3y-63 [illegible] 6055 p-10

AIAA-94-1209-CP
N94-30560

Object-Based Task-Level Control: A Hierarchical Control Architecture for Remote Operation of Space Robots

H.D. Stevens * E.S. Miles † S. J. Rock ‡ R. H. Cannon §
Stanford Aerospace Robotics Laboratory
Stanford, California 94305

Abstract

*† ‡§
Expanding man's presence in space requires capable, dexterous robots capable of being controlled from the Earth. Traditional "hand-in-glove" control paradigms require the human operator to directly control virtually every aspect of the robot's operation. While the human provides excellent judgment and perception, human interaction is limited by low bandwidth, delayed communications. These delays make "hand-in-glove" operation from Earth impractical.

In order to alleviate many of the problems inherent to remote operation, Stanford University's Aerospace Robotics Laboratory (ARL) has developed the Object-Based Task-Level Control architecture. Object-Based Task-Level Control (OBTLC) removes the burden of teleoperation from the human operator and enables execution of tasks not possible with current techniques. OBTLC is a hierarchical approach to control where the human operator is able to specify high-level, object-related tasks through an intuitive graphical user interface. Infrequent task-level commands replace constant joystick operations, eliminating communications bandwidth and time delay problems. The details of robot control and task execution are handled entirely by the robot and computer control system.

The ARL has implemented the OBTLC architecture on a set of Free-Flying Space Robots. The capability of the OBTLC architecture has been demonstrated by controlling the ARL Free-Flying Space Robots from NASA Ames Research Center.

1.0 Introduction

As NASA expands America's presence in space, on-orbit assembly, maintenance, and servicing must become routine operations. The extreme cost and risk of astronaut EVA dictate that automation and robotics must play a key role in providing such services in any viable long-duration human-in-space future. The enormous number of EVA hours currently required to perform these operations can be significantly reduced by the timely provision of effective human/robot teams. Such a team would consist of a human in a safe haven, such as on Earth or inside a space vehicle, indicating at a high level the tasks to be done, while robots in the space environment execute the tasks with quick proficiency.

To date, the operation of space robots requires the user to manually control the robot's actions *directly* by a "hand-in-glove" method (i.e. teleoperation). Robot performance is consequently characterized by the *fundamental* limitations of any system where human control is intricately involved -- namely,

* Ph.D. Candidate, Department of Aeronautics and Astronautics. Member AIAA. hdsteven@sun-valley.stanford.edu

† Ph.D. Candidate, Department of Aeronautics and Astronautics. esm@sun-valley.stanford.edu

‡ Associate Professor, Department of Aeronautics and Astronautics. Member AIAA. rock@sun-valley.stanford.edu

§ Charles Lee Powell Professor, Department of Aeronautics and Astronautics. cannon@sun-valley.stanford.edu time delay between the human and robot due to long distance communications, low bandwidth performance due to slow human response characteristics, and intense operator tedium and fatigue due to the complexity of teleoperating complex dynamic systems. Clearly, these limitations call into question the viability of teleoperated systems for the extended, sophisticated on-orbit operations for which they are intended.

Object-Based Task-Level Control (OBTLC), an architecture developed by Stanford University's Aerospace Robotics Laboratory (ARL), removes the burden of teleoperation from the human operator, enabling execution of tasks not possible with current teleoperation techniques. OBTLC is a hierarchical approach to control where the human operator is able to specify high-level, object-related tasks through an intuitive graphical user interface. Occasional task-level commands replace constant joystick operations, eliminating communications bandwidth and time delay problems. The details of robot control and task execution are handled entirely by the robot and computer control system.

2.0 THE OBTLC ARCHITECTURE

In order to fully comprehend the OBTLC architecture, it is first necessary to have a clear understanding of the terms "object" and "task", as they are used in this paper.

The notion of an object is fundamental to the OBTLC architecture. An object is any physical entity that the operator wishes to manipulate and/or to which a specific relationship with the environment or other objects is desired. An object might be something independent of the robot, such as an Orbital Replacement Unit (ORU), a space truss member, a tool or a bolt; or it might be a significant part of the robot, such as a manipulator end-effector or perhaps the entire robot.

A task is integrally related to this notion of an object. Specifically, a task is a manipulation of objects in the environment (including robots) to match a desired configuration of, or relationship between, objects. Examples of tasks include: "replace that ORU with this ORU", "join these two truss members together", "extract that bolt with this wrench", and commanding a free-flying space robot to "move from point A to point B." In all of the above task examples, one theme is constant: task specifications directly correspond to high-level desired **object** behavior, not low-level details of robot manipulation and control to achieve these tasks. This approach to control is therefore referred to as **object-based control**, and the tasks performed are **object-based tasks**.

The objective of the Object-Based Task-Level Control (OBTLC) architecture is to provide the human operator with the ability to specify directly, and in a simple way, the object-based tasks he or she wishes to execute. The details of how these tasks are carried out are handled autonomously by the robot, and therefore do not burden the operator. Thus, the human is free to concentrate on high-level issues, such as devising strategies and solving problems, while the robot's computers perform the fast calculations necessary to close control loops precisely and autonomously. This novel approach exploits the complementary capabilities of robotic control and human decision-making to construct a powerful human/robot team.

Implementation of the OBTLC architecture provides numerous advantages over lower-level remote teleoperation. First, the detrimental effects of time delay are minimized because the human operator is eliminated from the low-level control of the robot. Task level commands from the human and responses from the robot need only occur at infrequent intervals.

Second, operator fatigue is significantly reduced because the human is no longer burdened with the low-level details of teleoperating a sophisticated dynamic system. Finally, this complementary division of labor between human and robot enables the human/robot team to perform more complicated tasks than is possible with traditional teleoperation approaches.

THE HIERARCHICAL NATURE OF OBTLC

OBTLC involves the management of three different kinds of information:
1) Infrequent communication between human and robot(s) about tasks to be performed.
2) Strategic information used by a robot or shared between several robots to break complicated tasks into smaller sub tasks.
3) Low-level dynamic control information used to close high-speed control loops on each robot.

The OBTLC control architecture is correspondingly divided into three layers-- the User Interface, the Strategic Controller, and the low-level Dynamic Controller.

The **USER INTERFACE** maintains and displays a world model, and receives desired changes to the state of the world from the operator. By manipulating iconic images of objects in this world model, the operator simply and intuitively instructs the robot to perform complex tasks. For example, insertion of the icon of one part into another is all that is necessary to instruct the robot system to perform all actions necessary to complete the insertion task.

The second layer, the **STRATEGIC CONTROLLER,** is based upon a finite state machine structure and embodies the logic and decision-making capabilities necessary for the robot to operate autonomously. Examples include path-planning, advanced manipulation and assembly of objects, and multiple-robot coordination. The Strategic Controller monitors changes in the state of the world, new commands from the human operator, and low-level sensor information, and uses this information to devise and execute new plans and to dictate changes in low-level control behavior. It is also this layer that identifies and sends to the user interface indications of events or problems that may require closer operator attention.

The third layer, the **DYNAMIC CONTROLLER**, incorporates high-bandwidth, sensor-based feedback control to achieve precise, high-speed dynamic performance of the robot system. This layer renders all details of robot control (i.e. position and force regulation, coordination of dynamic coupling, use of redundancy, control optimization, disturbance rejection, etc.) completely transparent to the human operator.

3.0 RELATED WORK

There are several control Architectures designed for space operation. Lumia and Albus proposed the NASA/NBS Standard Reference Model for the Telerobot Control System Architecture (NASREM)[1]. NASREM is made up of three six-level hierarchies for task decomposition, world modeling, and sensory processing. In the NASREM system, the concept of an object at a high level is lost. The architecture is focused on controlling and coordinating manipulators. Strategic control, as defined in the previous section, is not incorporated into the NASREM architecture.

The Modular Telerobot Task Execution System (MOTES) [2], developed at JPL, is another type of hierarchical robot controller. The MOTES system is based on a command interpreter similar to that used in spacecraft. This approach differs from OBTLC in that it only generates plans that sequence pre-programmed,

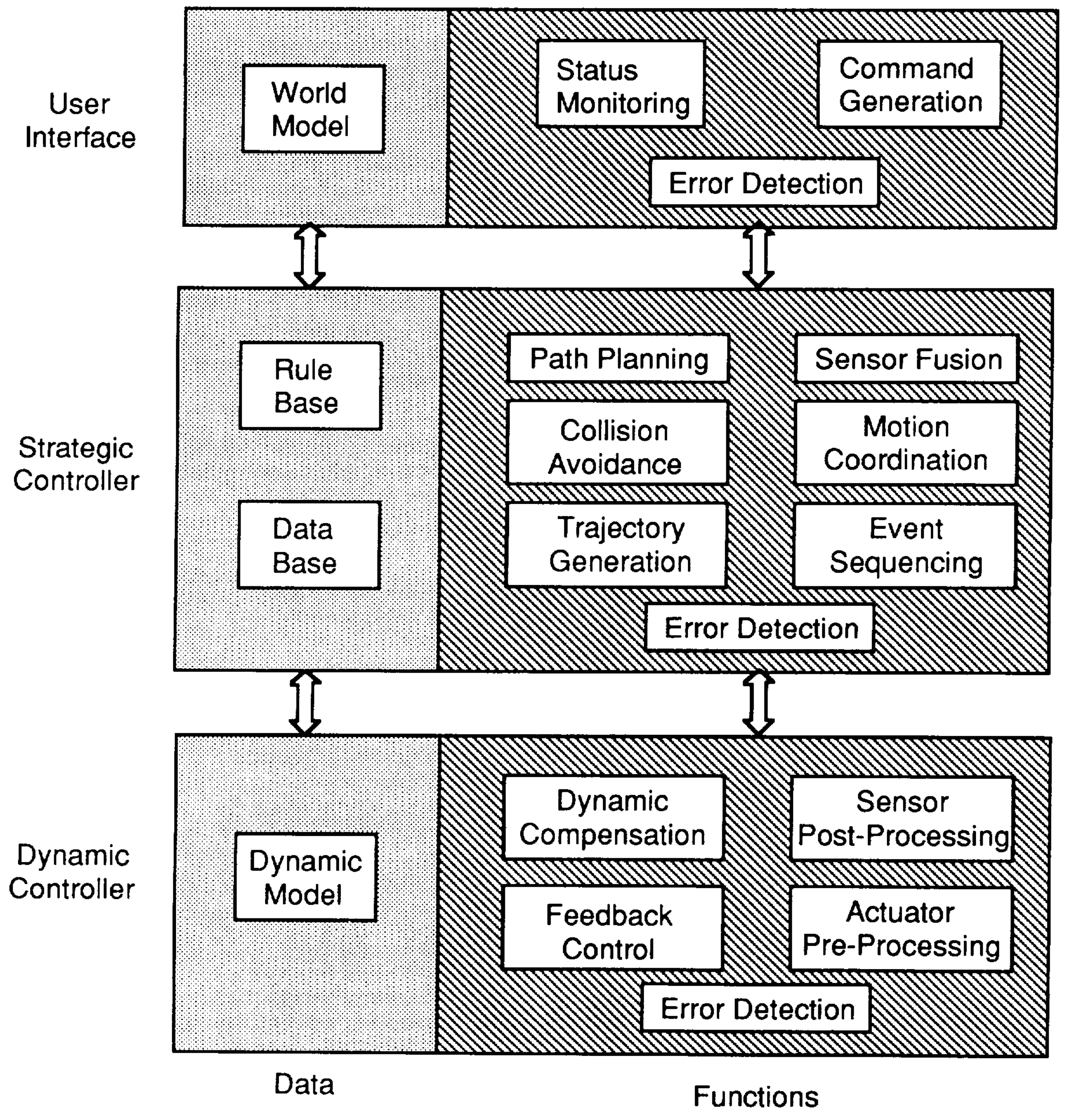


Figure 1: The Object-Based Task-Level Control Architecture. The architecture consists of a user interface, a strategic controller, and a dynamic controller. Occasional task-level commands from the user interface to the strategic controller create a system that is unaffected by communication delay.

open-looped macros and does not incorporate any sensor based decision making.

Another architecture which bears greater similarity to OBTLC is Sheridan's concept of supervisory control [3]. Indeed, at their most simplified level, both supervisory control and OBTLC involve human instructions to complex systems, which are than translated into actuator commands. In practice, however, most researchers interpret supervisory control to mean computer-augmentation of human teleoperation (i.e. incorporating control loops and compensators in the system to make teleoperation more tractable). OBTLC differs from this interpretation in that the human input to the system is at a much higher level; in fact, human input is absent at the lowest level. OBTLC therefore represents an exploration of Sheridan's concept in a novel direction.

4.0 IMPLEMENTATION OF OBTLC ON A FREE-FLYING SPACE ROBOT

OBTLC has been implemented on several experimental systems at Stanford ARL, including several mobile and stationary robots with cooperating manipulators [4,5,7,8,10], and an underwater vehicle [6]. The application of OBTLC to a free-flying space robot prototype [7,8] is particularly interesting because of the complexity of the system.

ARL's space robotics facility features three autonomous self-contained free-flying space robots. A space environment is simulated in two dimensions using an air bearing over a flat granite surface plate. In this environment, the robots float on a cushion of air approximately 0.003 inches thick, and they propel themselves using on/off compressed air thrusters. The space robot is equipped with an on-board compressed gas supply, two two-link SCARA configuration manipulators, an on-board power supply, on-board computing, wireless ethernet communications, and local vision-sensing capability.

These space robots are capable of a variety of tasks including: capturing a translating, spinning object, adaptively identifying an objects mass and inertia properties, cooperatively maneuvering large objects, and assembling multiple objects. All of the space robots are based on the Object-Based Task-Level Control paradigm, although each implementation is slightly different. In this manner, the OBTLC architecture continues to evolve in response to new requirements.

EXAMPLE TASK: CAPTURE THAT OBJECT

To fully explore the concepts involved in OBTLC, one should examine, in detail, what is involved in carrying out a specific task. The task of capturing a translating, spinning object with a free-flying space robot is a particularly good example. An object, called Scooter, floats on the same granite table as the robot and is not within the initial workspace of the robot's manipulators. The operator wishes to capture Scooter, necessitating that the robot rendezvous with and grasp Scooter. Figure 2 shows the robot and object.

A global sensing system provides position and orientation information for the objects on the table (i.e. the robot and Scooter) in real-time. This information is used by both the user interface and the strategic controller to update the world model.

USER INTERFACE

One implementation of the user interface uses the Virtual Environment Vehicle Interface (VEVI) developed by the Intelligent Mechanisms Group at NASA Ames Research Center. The VEVI is an interactive virtual reality user interface which utilizes real-time interactive 3D graphics and position/orientation sensing to produce a range of interface modalities from flat-panel (windowed or stereoscopic) screen displays to head mounted/head-tracking stereo displays [9].

The VEVI displays the virtual reality model of the world (robot, Scooter, and table) with the position and orientation of the objects updated at about 1 hz from the global sensing system. The operator simply manipulates the objects by controlling a virtual hand icon with a 3D mouse. To command a capture, the operator places the hand in select mode (using a button on the mouse) and

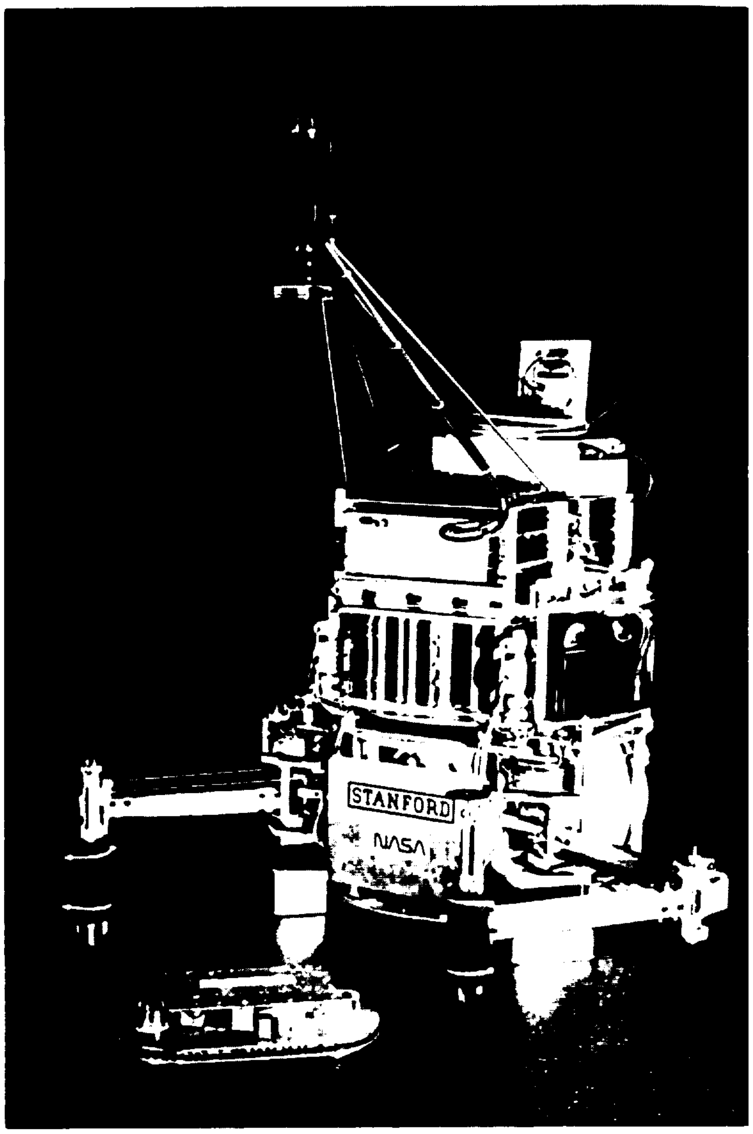
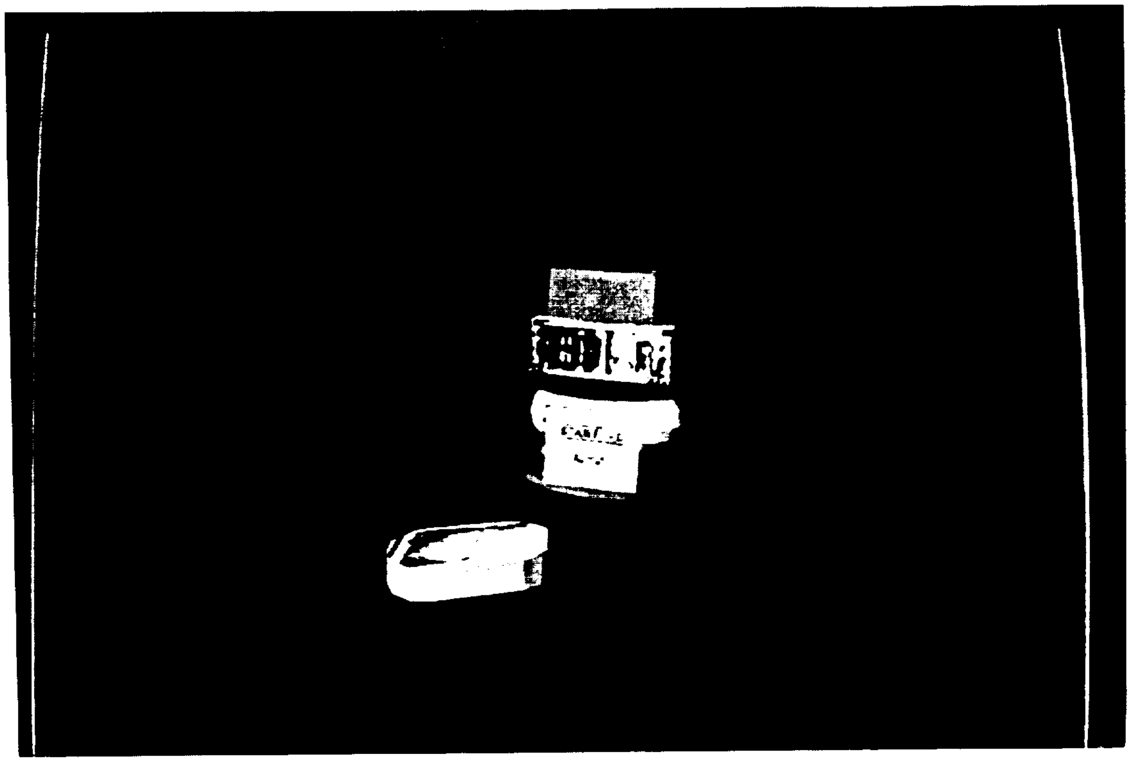


Figure 2: A Free-Flying Space Robot and an object. The space robot uses the Object-Based Task-Level Control architecture. ARL's Free-Flying Space Robots are capable of a variety of tasks including: capturing a translating, spinning object, adaptively identifying an objects mass and inertia, cooperatively maneuvering large objects, and assembling multiple objects.

Figure 3: The Virtual Environment Vehicle Interface. The VEVI, developed at NASA Ames Research Center, provides a simple, intuitive operator interface. By manipulating iconic images of the objects, the operator simply and intuitively instructs the robot to perform complex tasks.



Operation	Control Mode	Trajectory	Error Law
System Initialization	Joint	Fifth Order	PD
Rendezvousing with Object	Endpoint (Base Relative)	Set Point	PD
Intercepting Object	Endpoint (Inertial)	Fifth Order	PD
Tracking Object	Endpoint (Inertial)	Tracking	PID
Stopping Object	Object-Based (Base Relative)	Fifth Order	PD
Holding Object	Object-Based (Base Relative)	Set Point	PD
Placing Object	Object-Based (Inertial)	Fifth Order	PD

Table 1: Control Modes Required for Object Capture.
This table lists the set of controller configurations that the strategic controller takes the system through in the process of rendezvousing with and capturing a free-flying object. In all of these configurations, the base motion is controlled in the inertial reference frame using bang-off-bang trajectories and PD error laws.

touches the object. The VEVI then transmits the capture command, which requires the object name, Scooter in this case, as the only parameter. Figure 3 shows an operator's view of the VEVI.

It is this high-level of interaction that enables low-bandwidth communication and eliminates the effect of time delay. The operator is now free to plan the next task, contemplate the strategy, or just watch the task execution.

ON THE ROBOT: STRATEGIC CONTROLLER and DYNAMIC CONTROLLER

Upon receipt of the capture command the strategic controller begins a multi-step process of intelligently carrying out the capture task. The strategic controller is implemented using a finite state machine. As new events or stimuli occur, the finite-state machine reacts, depending on the current state of the system, by either progressing to the next phase of a multi-step procedure or by initiating a new course of action.

A major portion of the strategic controller's coordination involves the switching of control modes in the dynamic controller. There are seven different control modes required to complete the capture task. These seven are listed in Table 1. All of these are implemented in the dynamic controller. A complete discussion of these low-level control modes can be found in [8]. One control mode of interest is the object-based control mode. This control mode is based on the theory of Object Impedance Control [4,10]. This control methodology carries the concept of object down to the lowest levels of the control architecture.

The capture command sets in motion the finite state machine (FSM) to capture the object. The topology of the FSM is depicted in Figure 4. In the figure an ellipse signifies a state in the finite state machine, a rectangle signifies a state transition procedure, and a phrase over a line indicates the stimulus that causes the transition from one state to another. State transition procedures are similar to subroutines that return a stimulus. Thus each procedure completes some actions and returns the appropriate stimulus.

To complete the capture, the strategic controller determines if the object requested is either in view (i.e. within the range of the local sensing system) or found (i.e. on the table, but not within view of the local sensing system). In the example, the object is found. The object trajectory and robot base intercept trajectory are computed and the proper dynamic controllers switched in. The dynamic controller is provided with the proper intercept trajectory to follow. At

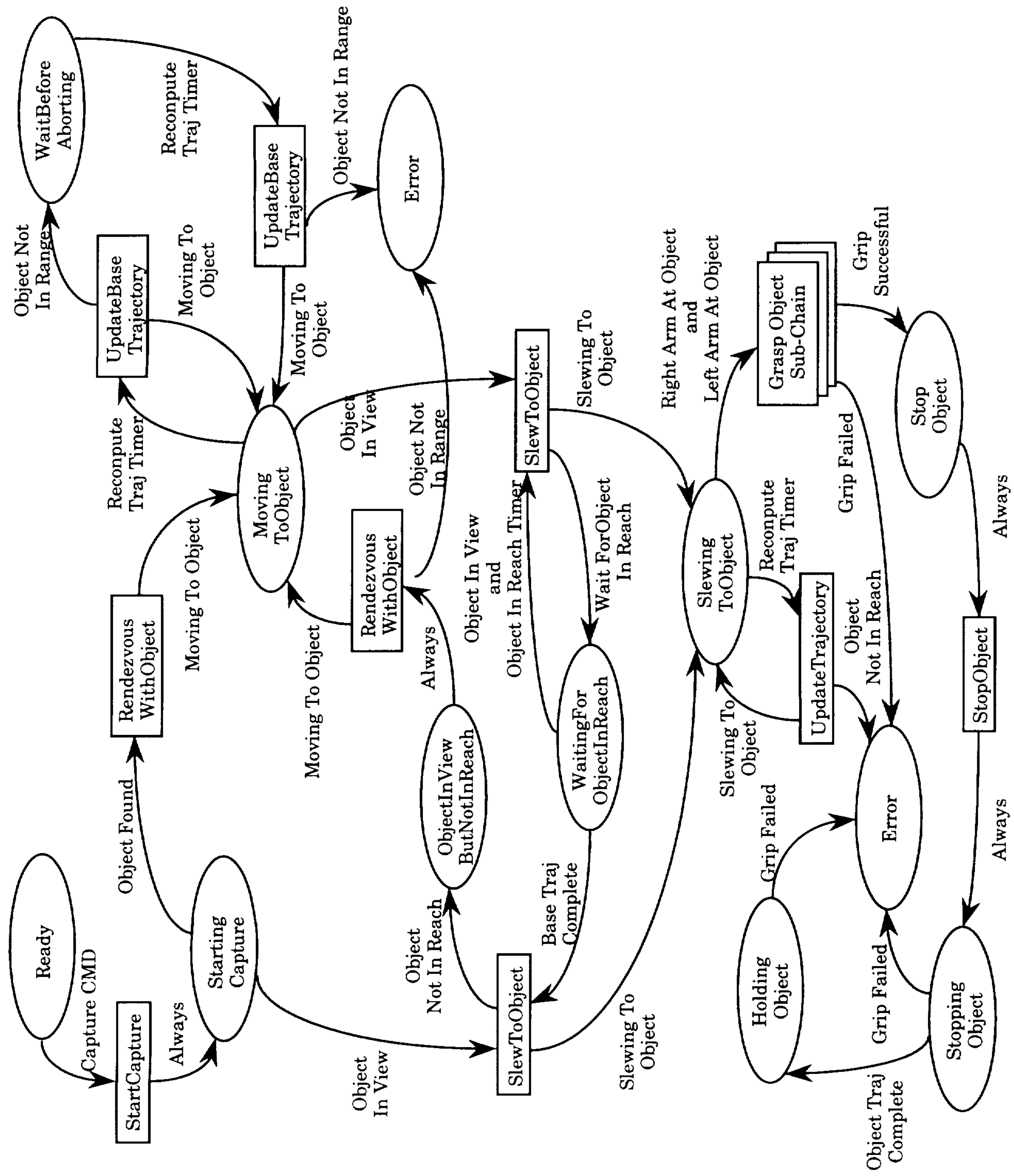


Figure 4: Object Rendezvous Finite-State Machine Graph.
This is the portion of the Strategic Controller which is executed when a Capture command is issued by the operator. Using the Finite-State Machine, the Strategic Controller is able to react, intelligently, to new sensor information. This sensor based decision ability is the unique feature of OBTLC.

regular intervals, the intercept trajectory is recalculated to allow for new information to enter the system. The base motion and trajectory recalculation continue until the object comes into view of the local sensing system.

With the object in view of the local sensing system, the robots manipulators are commanded to begin slewing to the object. Trajectories for each of the two manipulators are computed, checked for collisions, and executed as the object comes within the workspace of the manipulators. The trajectories place the endpoints over the grip points for grasping. The object is grasped, and the motion of the object stopped using the manipulators. Scooter has been successfully captured.

The entire sequence of events described above is initiated with a simple "capture that object" command issued by the operator. The operator has been completely removed from the details of robot motion and control modes required to complete this capture. It is apparent that the details of this operation, and the speed at which they must be accomplished, are daunting for the human operator alone. It is quite possible that a human operator with no help could not even accomplish this task.

5.0 CONCLUSIONS

The OBTLC architecture is a powerful new paradigm in the remote control of robot systems where the operator interacts with the system via an intuitive interface. The system is commanded at the task level, allowing the human operator to focus on the strategic issues, such as what to do next, while the robot system carries out the desired tasks quickly and deftly. This paradigm raises the human/robot team to a level never before possible.

Development of the OBTLC architecture has been guided by the principles of systems engineering and the desire to enable humans to interact with a robotic system at an intuitive level. This architecture has evolved to the current point only by the strict adherence to these principles. As with any architecture, OBTLC continues to evolve enabling its application to a broad range of problems.

6.0 ACKNOWLEDGMENTS

This research is supported by NASA Grant NCC 2-333. The authors are grateful to NASA for their continued support of this research.

7.0 REFERENCES

[1] Ronald Lumia and James S. Albus. Teleoperation and Autonomy for Space Robotics. *Robotics,* 4(1):27-33, 1988.

[2] Paul G. Backes, Mark K. Long, and Robert D. Steele. The Modular Telerobot Task Execution System for Space Telerobotics. In *Proceedings of the IEEE International Conference of Robotics and Automation*, pages 524-530, Atlanta, GA, May 1993. IEEE Robotics and Automation Society.

[3] Thomas B. Sheridan. Telerobotics, Automation, and Human Supervisory Control. Cambridge, Massachusetts: The MIT Press, 1992.

[4] Stanley A. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators*. PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, September 1989. Also Published as SUDAAR 586.

[5] Lawrence E. Pfeffer. *The Design and Control of a Two-Armed, Cooperating, Flexible-Drivetrain Robot System.* PhD thesis,

Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, December 1993. To Be published.

[6] Howard H. Wang, et all. Task-Based Control Architecture for an Untethered, Unmanned Submersible. In Proceedings of the 8th Annual Symposium of Unmanned Untethered Submersible Technology, September 1993.

[7] Marc A. Ullman. *Experiments in Autonomous Navigation and Control of Multi-Manipulator, Free-Flying Space Robots*. PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, March 1993. Also published as SUDAAR 630.

[8] William C. Dickson. *Experiments in Cooperative Manipulation of Objects by Free-Flying Robot Teams*. PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, December 1993. To be published.

[9] T. W. Fong. A Computational Architecture for Semi-autonomous Robotic Vehicles. In *Proceedings of the AIAA Computing in Aerospace 9 Conference,* San Diego, CA, October 1993. AIAA.

[10] Stanley A. Schneider and Robert H. Cannon, Jr. Object impedance control for cooperative manipulation: Theory and experimental results. *IEEE Journal of Robotics and Automation*, 8(3). June 1992.

6056

AIAA-94-1210-CP

Task-Level Control for Autonomous Robots

N94-30561

Reid Simmons
School of Computer Science / Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
reids@cs.cmu.edu

p 7

Abstract

Task-level control refers to the integration and coordination of planning, perception, and real-time control to achieve given high-level goals. Autonomous mobile robots need task-level control to effectively achieve complex tasks in uncertain, dynamic environments. This paper describes the Task Control Architecture (TCA), an implemented system that provides commonly needed constructs for task-level control. Facilities provided by TCA include distributed communication, task decomposition and sequencing, resource management, monitoring and exception handling. TCA supports a design methodology in which robot systems are developed incrementally, starting first with deliberative plans that work in nominal situations, and then layering them with reactive behaviors that monitor plan execution and handle exceptions. To further support this approach, design and analysis tools are under development to provide ways of graphically viewing the system and validating its behavior.

Introduction

Most autonomous robot systems have specific tasks to perform — such as navigating to given locations, searching for particular objects, exploring the environment, etc. To make a robot perform its tasks reliably, it is desirable to provide explicit control over the achievement of tasks — controlling the sequencing of subtasks, monitoring their progress, handling exceptions, and managing the robot's limited computational and physical resources.

We refer to this as *task-level control:* the integration of planning, perception, and real-time control for the purpose of achieving high-level goals. To facilitate the development of task-level control systems, we have developed the Task Control Architecture (TCA). To date, TCA has been used in the development of about a dozen autonomous robot systems, including a walking rover [Simmons *et al.*, 1992], several indoor mobile robots [Simmons *et al.*, 1990], an excavator [Singh and Simmons, 1992], and an inspection robot for the Space Shuttle [Dowling and others, 1992].

The motivation for developing a task-level control architecture is that there appears to be a common set of control constructs that most autonomous mobile robots need, and that development of individual robot systems can be simplified by use of an architecture that explicitly supports those constructs. In much the same way as an operating system provides common facilities and hides details of the underlying computer, so too does TCA provide needed task-level control constructs while hiding details such as the mechanisms used for communication and task synchronization.

The facilities provided by TCA were chosen based on analysis of existing mobile robot systems and projected needs of future systems. The analysis showed that the architecture must facilitate the development of distributed, modular, and concurrent systems. In addition, a task-level control architecture should allow the concurrency to be controlled in a selective (and explicit) manner, so that distributed processes do not interact in undesirable ways. This includes providing methods for sequencing and synchronization of subtasks, as well as managing access to system resources (e.g., cameras, actuators, computers). Finally, to cope with uncertainties in the environment and uncertainties in the achievement of subtasks, the architecture needs to support extensive, task-dependent monitoring and exception-handling strategies.

In addition to providing all the above capabilities, the Task Control Architecture supports a particular methodology for designing and developing autonomous robot systems. The approach, which we term *structured control*, involves first developing basic deliberative components that handle nominal situations, and then increasing reliability by incrementally layering on reactive behaviors to handle exceptions. With TCA, this can be done without requiring significant modification to the existing robot software system. In particular, monitors and exception handlers can be added after the

[1]Copyright ©1993 by Reid Simmons. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

basic system has been developed.

This layering of reactive behaviors on to a deliberative base provides an engineering basis for developing autonomous mobile robot systems. First, incomplete understanding of the tasks, environment or hardware is accommodated by separating the design into nominal, and presumably better understood, behaviors and the more numerous, but infrequently occurring, exceptional situations (which may become known and understood only during testing of the robot system). Second, the separation of nominal and exceptional behaviors increases overall system understandability by isolating different concerns: the robot's behavior during normal operation is readily apparent, and strategies for handling exceptions can be developed separately and then added to the existing system with a minimum of effort. Finally, complex interactions are minimized by constraining the applicability of reactive behaviors to specific situations, so that only manageable, predictable subsets of the behaviors will be active at any one time.

The rest of this paper describes the Task Control Architecture in more detail, focusing on a few applications of the architecture to the development of autonomous mobile robot systems. The paper concludes with a brief description of where the development of TCA is heading — in particular, describing design and analysis tools that we are beginning to develop.

The Task Control Architecture

The Task Control Architecture has been designed to facilitate the process of developing and controlling autonomous robot systems that must perceive, plan and act in uncertain, dynamic environments [Simmons, 1992a, Simmons, 1992b]. TCA provides a language for expressing task-level control decisions, and provides software utilities for ensuring that those control choices are correctly realized by the robot. The five major types of control constructs supported by TCA are:

- distributed communication
- task decomposition and sequencing
- resource management
- execution monitoring
- exception handling

A system built using TCA consists of robot-specific processes (called ***modules***) that communicate by sending messages via a general-purpose *central control module* (see Fig. 1). Modules can be written in either C or LISP, and can operate on a number of different computer platforms (including Sun, SGI, Vax, MacIntosh, and 680xx and i486 processors) and on different operating systems (including Unix, VxWorks and Mach).

The robot-specific modules register with the central control module which messages they can handle, along with the data formats associated with the messages. The data formats can be complex, including embedded structures, arrays, and pointers. TCA is responsible for encoding and decoding the data into byte streams and routing messages (via sockets) to the appropriate

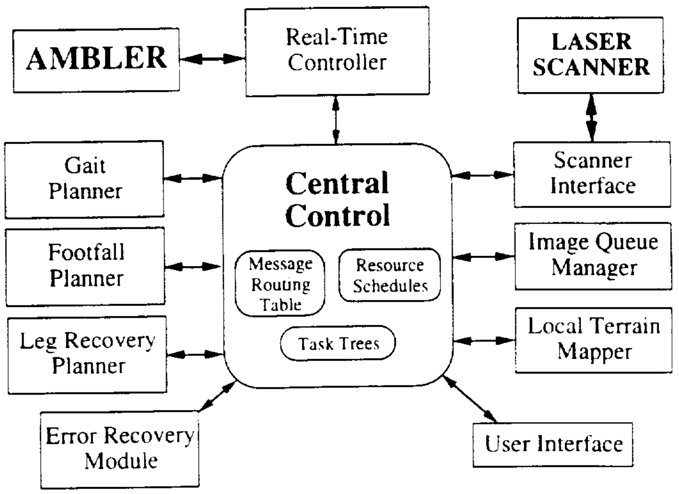


Figure 1: Task Modules for Ambler Walking System

modules to be handled. Messages are ***anonymous***, that is, the sending and receiving modules do not know each other's identities. This facilitates modular development — one module can easily be substituted for another with the same functionality (even while the rest of the system continues to operate). Thus, for example, a graphical simulator that has the same message interface as the real-time controller can be substituted at will, which greatly facilitates the development and debugging process (as well as protecting valuable robotic hardware!).

TCA provides different types of messages, each with somewhat different semantics. For example, *inform* messages provide one-way communication between processes; *query* messages provide two-way communication (providing a client-server relationship), and *broadcast* messages enable one module to distribute data to any number of receiving modules simultaneously. Other message types, including goals, commands, monitors and exceptions, will be discussed below.

Task Decomposition

Besides providing for data communication, TCA provides a host of facilities for coordinating robot systems at the task level. Modules use the TCA control constructs to constrain the robot's behavior. For example, a module can specify the order in which subtasks should be carried out, or indicate when and how to monitor for exceptional conditions.

Central to TCA is a hierarchical representation of subtasks called *task trees*. In essence, a task tree is TCA's notion of a plan, representing both goal/subgoal decomposition, as well as *temporal constraints* between node, which indicate (partial) orderings on their execution. TCA constructs and maintains task trees dynamically: nodes in the task tree are associated with

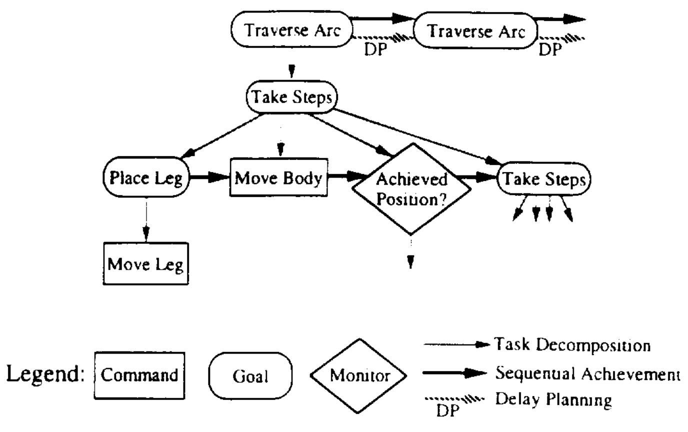


Figure 2: Task Tree for Ambler Walking

messages; when a message handler itself issues a message, a child is added under the node associated with the message being handled. TCA utilizes the subgoal and temporal constraint information to schedule and coordinate the sending of messages.

Figure 2, for instance, illustrates a simplified version of the task tree for autonomous walking of the Ambler rover [Simmons *et al.*, 1992]. In the figure, narrow vertical arrows denote task decomposition and heavy horizontal arrows denote temporal constraints on task planning and execution. The task tree indicates that the Ambler sequentially traverses a series of arcs, where planning how to traverse one arc is delayed until the previous arc has been completely traversed. Traversing an arc consists of taking a sequence of steps, with each step consisting of a pair of leg and body moves. Unless the end of the arc has been reached, the planning module handling the "Take Steps" message recursively issues another "Take Steps" message. Note that the absence of a *delay planning* (DP) temporal constraint between the "Achieved Position?" monitor node and subsequent "Take Steps" goal node indicates to TCA that planning one step can occur concurrently with the execution of the previous step. This use of concurrency enables the Ambler to achieve nearly continuous motion [Simmons, 1992a].

Resource Management

Many robot systems have limited resources that must be managed efficiently. This is particularly important when the robot system consists of multiple, interacting processes in order to prevent resource contention and conflict. For example, if the robot has a camera on a pan/tilt head, the processes that need visual information must have ways to point the camera and to ensure that no other process will re-aim it until the required images have been acquired. Similarly, a robot system might want to ensure that a planning module remained available to deal with an upcoming, high priority request.

TCA provides support for this type of resource management. Procedures that handle messages can be grouped into logical units, called *resources*. These units can, in turn, be grouped into modules (see, for instance, Fig. 1). TCA maintains the constraint that only one message will be handled by a resource at a time. However, since modules may consist of multiple resources, a module can be processing multiple messages at once (for instance, if it is running in a multi-tasking environment such as VxWorks). This division into resources and modules is totally up to the discretion of the robot system designer, and can be organized so as to promote modularity, efficient use of resources, or the need to access a common piece of hardware.

TCA also enables a module to *lock* the resource of another module. This prevents any other module from accessing the resource until it is unlocked. This provides a mechanism for synchronizing subtasks: the resource can be locked while a time-critical operation is taking place, and then unlocked to enable normal message flow. In the Ambler system, for example, the perception module locks the real-time controller resource before acquiring laser range images, in order to prevent blurring.

Monitoring and Exception Handling

One of the most important task-level control functions for an autonomous mobile robot is to monitor its progress and safety, and to handle exceptions arising from violated expectations. The structured-control approach to designing robot systems advocates that such reactive behaviors be added incrementally, on top of the task tree that represents the basic, deliberative plan for achieving the task.

The rationale here is that, for complex tasks and environments, it is too difficult to design a system from the start that acts correctly in all situations. This is primarily because either the environment is not that well understood (especially if it is dynamic or remote, such as the surface of another planet) or the interactions between the environment and the robot are not well understood (such as for an excavation robot). Often the best that can be done in such cases is to design for the known situations first, and then incrementally debug and extend the system as experience dictates.

TCA provides several mechanisms that directly support this approach. For one, exception handling strategies can be added incrementally without modifying existing components: a module can add information to an existing task tree to indicate which procedures TCA should invoke in response to exceptions raised by other modules. When an exception is raised, TCA searches up the task tree to find a handler designated for that exception. If the exception handler finds it cannot actually deal with the particular situation, it reissues the exception and the search continues up the tree. Typically, the strategies for dealing with exceptions involve

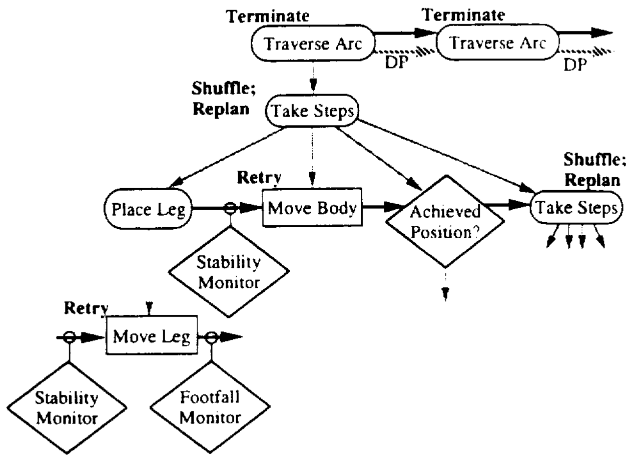


Figure 3: Task Tree with Monitors and Exception Handlers

modifying the currently executing plan, either by killing off parts of the task tree or adding new nodes and/or temporal constraints to the tree.

For example (Fig. 3), the Ambler real-time controller monitors force sensors in the feet and raises an exception when a threshold is exceeded (indicating unexpected terrain contact). A separate error recovery module handles this by modifying the current leg trajectory to surmount the obstacle, and then instructs TCA to re-execute the trajectory [Simmons, 1992b]. If modifying the leg trajectory fails to clear the obstacle, the complete move may be replanned, the Ambler's feet may be shuffled into a standard configuration, etc. Ultimately, if no fix is found, the walking task is terminated and the user is notified.

Just as it makes sense to take advantage of hierarchy in decomposing tasks into subtasks, it makes sense to treat exceptions in a hierarchical fashion. The idea is that lower-level exception handlers are more specific to a given failure, and can have more local, direct effects on the problem; the handlers located higher up the tree handle a wider range of exceptions, but since their effects are broader and have more impact on the overall plan, they should be tried only when the more specific strategies fail.

Execution monitors can also be added incrementally using the TCA *wiretap* control construct. The wiretap mechanism enables a monitor to be associated with a class of messages, so that the monitor is automatically triggered whenever a message of that class is handled. For example, before every leg or body move of the Ambler, a stability monitor is invoked to verify that the move will not cause the robot to tip over; after every leg move, a footfall monitor analyzes the force sensor data to detect possibly unstable footholds (see Fig. 3).

These monitors were added after the basic walking component of the Ambler was designed and debugged, in order to enable the system to handle increasingly difficult terrain and longer distances. For example, in one experiment, the Ambler walked over 500 meters outdoors in hilly terrain (with slopes up to 30%). During the experiment, in which the Ambler took over 1000 footsteps, many exceptional situations were encountered: unexpected terrain collisions, hardware faults (amplifiers, motion faults, sensor failures) and software faults (mainly when the planners could not find suitable footfalls). All these situations were dealt with by the robot itself: the conditions were detected in a timely manner and, except for certain hardware faults where humans had to manually reset the hardware, the robot autonomously recovered from the situations and continued walking.

Monitors can also be added to check for ongoing opportunities or contingencies. For example, one of our indoor mobile robots has the task of keeping the lab floor free of cups [Simmons *et al.*, 1990]. The robot system employs one monitor to check whether a new cup has been spotted by the vision system. For every cup found, a goal is added to retrieve the cup and another monitor is added which checks to ensure that the cup is still visible. If the cup disappears from view, then it is assumed that someone else picked it up, and the monitor cancels the associated "cup retrieval" task. Thus, the system is able to handle multiple goals that are both activated and deactivated asynchronously.

Comparisons

TCA and the structured-control approach differ from the *behavior-based* approach, in which systems consist of collections of local behaviors that act according to direct sensing of the environment [Brooks, 1986, Connell, 1989]. The global behavior of such systems typically emerge from interactions between the local behaviors [Agre and Chapman, 1987, Brooks, 1991]. A problem with the behavior-based approach is it assumes that robust primitive behaviors can be developed that act correctly in all, or most, situations. This can be very difficult in practice, given incomplete knowledge about the environment and the robot's interaction with it. In contrast, the structured-control approach advocates developing complete components for limited environments, and incrementally updating the design to handle more challenging and diverse requirements.

The approach also differs from other hierarchical architectures, such as NASREM [Albus *et al.*, 1989], in which the flow of control is primarily top-down. While top-down task decomposition is important in TCA, the architecture also provides for significant bottom-up control in its use of monitors and hierarchically scoped exception handlers. This enables autonomous robot systems to be very reactive to changes in the environment.

The approach used in TCA is probably closest in flavor to the RAPs system [Firby, 1989] and related architectures [Gat, 1992, Georgeff and Lansky, 1987],

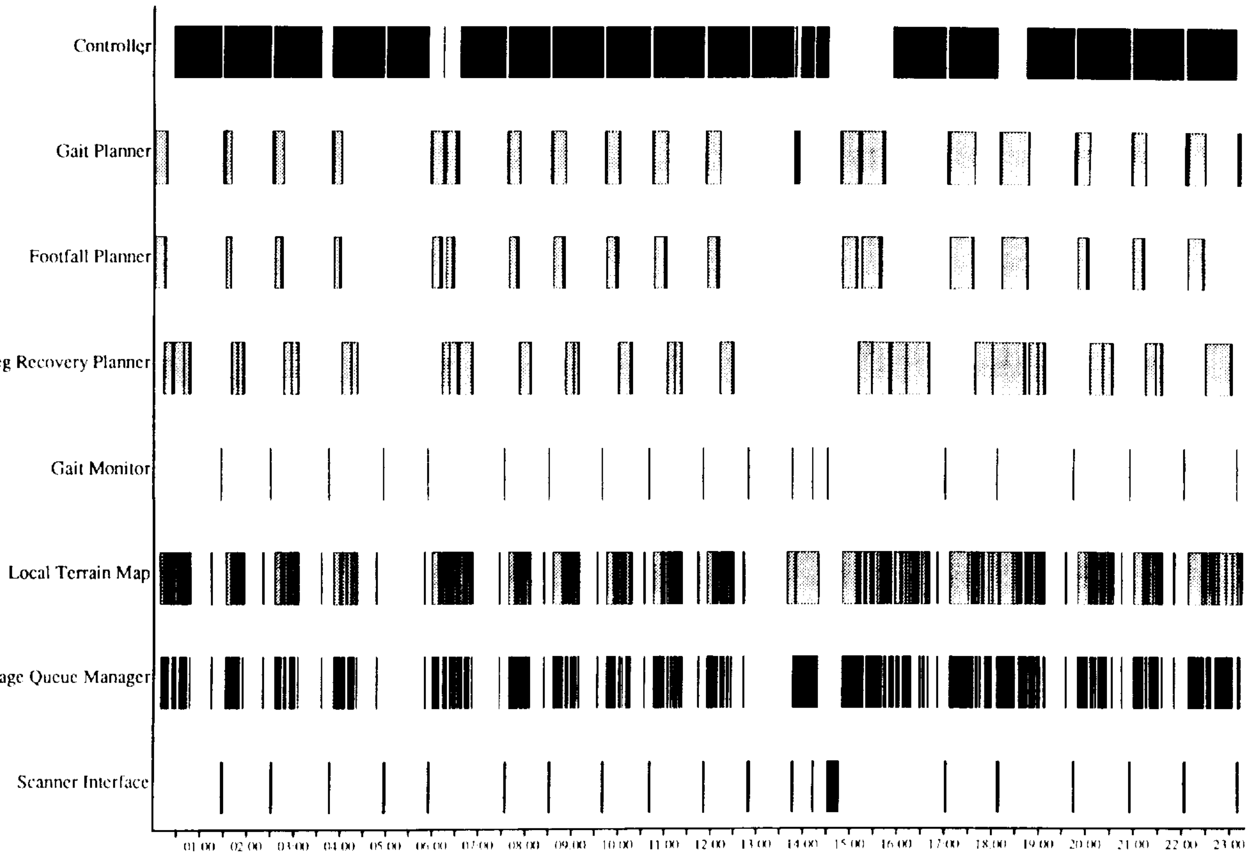


Figure 4: Gantt Chart of Module Activity

which feature temporal sequencing of subtasks in conjunction with monitoring and error recovery. The main differences are that TCA is based on true concurrency, rather than interleaving of subtasks (which allows it to exhibit better real-time performance), and that planning, monitoring and exception handling are all cleanly separated (which facilitates evolutionary robot system development).

Design and Analysis Tools

While TCA and the structured-control approach have proven useful for complex, autonomous robot systems, in practice developing such systems is often a time-consuming, trial-and-error process. To reduce this effort, we are currently developing tools to aid in the analysis and design of TCA-based robot systems.

The first two tools that we developed analyze the log files that TCA produces of all message traffic. The log files contain important information regarding the types and order of messages sent within the system. One tool in current use processes log files and produces graphical representations of TCA task trees (similar to that shown in Fig. 2). A developer can recreate the task tree message by message, either *post hoc* or as the system runs, to see what the task tree looks like as it evolves, and what temporal interactions might be causing problems. This tool has proven particularly valuable because it is typically difficult to predict in advance the behavior of complex distributed systems due to subtle timing interactions between processes.

Another tool analyzes log files to produce Gantt charts showing module activity (see Fig. 4 — the dark bars indicate when a module is processing messages; the light bars indicate when it is waiting for the reply to a query message). For each module, the chart shows which messages it is processing at what times, and when messages are queued due to resource contention. This tool has been used to find bottlenecks in system performance. For example, it was used in the development of the Ambler system to determine how to maximize performance through the use of concurrency. The Ambler system was originally developed with a sequential sense-plan-act cycle. The use of this tool indicated that continuous motion could be obtained by executing one step while planning the next one, since the time needed for executing steps exceeded the planning time for steps [Simmons, 1992a]. More recently, a similar analysis indicated that perception was the bottleneck in system performance: based on this, TCA control constructs were added to make some of the perceptual processing concurrent, as well [Hoffman and Krotkov, 1991].

We are beginning development of additional tools to aid in the design of mobile robot systems. One tool, similar in spirit to a CASE tool, would enable designers to graphically specify task decomposition strategies, including conditionals, loops, temporal constraints, monitors and exception handlers. The tool would then generate the TCA calls needed to implement those specifications. We anticipate that this tool will be very useful in rapidly prototyping system designs and in documenting the design process.

Eventually, we would like for the tool to actually help validate the system design, detecting problems such as malformed data interfaces between modules, potential deadlock situations, resource contention, etc. To do this, we need to apply automated reasoning techniques to TCA-based system designs (for instance, using model-checking techniques [Clarke *et al.*, 1986]). To this end, we have begun formalizing the Task Control Architecture control constructs using a combination of temporal logic and the Z notion [Spivey, 1992].

For example, the following schemas give the basic formalization of the notion of task trees: a task tree is a set of nodes, each of which has a parent. The "received" set consists of the messages that TCA has received and the "finished" set contains those that have already been handled by some module. A task tree node, in turn, has an associated handler, type, and state (received, running, finished) and a set of temporal constraints. The task tree schema places some conditions on the temporal constraints of various nodes of the task tree.

$$
\begin{array}{|l}
\textit{TaskTree} \\
\hline
\textit{nodes} : \mathbb{P}\, \textit{Node} \\
\textit{parent} : \textit{Node} \nrightarrow \textit{Node} \\
\textit{received} : \mathrm{seq}\, \textit{Node} \\
\textit{finished} : \mathrm{seq}\, \textit{Node} \\
\hline
\forall\, \textit{node}, \textit{node2} : \textit{Node} \bullet \\
\quad (\textit{node.type} \in \{\textit{Query}, \textit{Inform}\} \Rightarrow \\
\qquad \textit{parent}(\textit{node}) = \textit{root} \land \\
\qquad \textit{node.achievConst} = \varnothing \land \\
\qquad \textit{node.onHoldUntil} = \varnothing) \land \\
\quad (\textit{parent}(\textit{node}) = \textit{node2} \Rightarrow \\
\qquad \textit{node.achievConst} \subseteq \textit{node2.achievConst}) \land \\
\quad \textit{node.handler} = \textit{node2.handler} \Leftrightarrow \\
\qquad \textit{node} = \textit{node2} \\
\\
\textit{root} \notin \textit{nodes} \\
\\
\textit{nodes} = \mathrm{ran}\, \textit{parent} \land \mathrm{dom}\, \textit{parent} = \textit{nodes} \cup \{\textit{root}\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\textit{Node} \\
\hline
\textit{handler} : \textit{HANDLER\_ID} \\
\textit{type} : \textit{NODE\_TYPE} \\
\textit{state} : \textit{EXECUTION\_STATE} \\
\textit{achievConst} : \mathbb{P}\, \textit{TEMPORAL\_CONSTRAINT} \\
\textit{onHoldUntil} : \mathbb{P}\, \textit{TEMPORAL\_CONSTRAINT} \\
\hline
\textit{type} = \textit{Command} \Rightarrow \\
\quad \textit{achievConst} = \textit{onHoldUntil} \\
\\
\textit{type} \in \{\textit{Query}, \textit{Inform}\} \Rightarrow \\
\quad \textit{achievConst} = \varnothing \land \\
\quad \textit{onHoldUntil} = \varnothing \\
\hline
\end{array}
$$

When the formalizations are completed, we expect to use them to prove properties about the performance of specific robot systems. For example, using the current temporal formalization, we can show that the temporal constraints described in [Simmons, 1992a] are sufficient to ensure that the Ambler walking system will plan at most one step in advance. We would also like to use the Z formalization to prove the correctness of the implementation of TCA, to give users confidence that the architecture correctly meets the intended semantics.

Conclusions

Autonomous robot systems need task-level control in order to effectively integrate planning, perception and actuation to perform complex tasks in uncertain, dynamic environments. The Task Control Architecture (TCA) has been developed to facilitate the creation of task-level control systems. TCA provides control constructs that are commonly needed by autonomous robot systems, including distributed communication, task decomposition and sequencing, resource management, monitoring and exception handling,

TCA supports the *structured-control* methodology of system development in which plans are first designed to work in nominal situations, and then reactive behaviors (execution monitors and exception handlers) are layered on to the base of deliberative plans. We argue that such a design philosophy is useful in situations where the environment the robot will be operating in, and/or the robot/environment interactions, are not totally understood.

It is our contention that reliable performance in a wide range of situations can best be obtained by incrementally adding on reactive behaviors that deal with specific, previously unanticipated, situations. It is also beneficial to structure such behaviors hierarchically, relying first on lower-level reactions that have specific, but local, effects, and using higher-level reactions with more global effects only when the more specific ones fail to solve the problem.

TCA and the structured-control design methodology have been used in developing about a dozen autonomous mobile robots, including a planetary rover, an indoor mobile manipulator, an excavator, and a robot for inspecting the Space Shuttle. In each case, the communication and control constructs provided by TCA made it easier to develop and debug the concurrent, distributed systems.

We are continuing our efforts by providing design and analysis tools to support the development of TCA-based systems. In particular, we are formalizing the TCA control constructs in order to provide tools for automatically reasoning about and validating system designs.

Acknowledgments

Many thanks to Christopher Fedor, Long-Ji Lin and Richard Goodwin who implemented much of TCA, and members of the Planetary Rover project who have developed the Ambler and its autonomous walking system. Domingo Gallardo did the Z formalization of TCA. Thanks also to David Garlan for many useful ideas and discussions. This research is supported in part by NASA under Contract NAGW-1175 and by OSD under Contract F08635-92-C-0019.

References

[Agre and Chapman, 1987] Phil Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proc. National Conference on Artificial Intelligence*, pages 268–272, Seattle, WA, 1987.

[Albus *et al.*, 1989] James S. Albus, Harry G. McCain, and Ronald Lumia. NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical Report 1235, National Institute of Standards and Technology, 1989.

[Brooks, 1986] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), 1986.

[Brooks, 1991] Rodney Brooks. Intelligence without reason. In *Proc. International Joint Conference on AI*, Sydney, Australia, August 1991.

[Clarke *et al.*, 1986] Edward Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[Connell, 1989] Jonathan Connell. A behavior-based arm controller. *IEEE Journal of Robotics and Automation*, 5(6):784–791, 1989.

[Dowling and others, 1992] Kevin Dowling et al. Mobile robot system for ground servicing operations on the space shuttle. In *SPIE, Cooperative Intelligent Robotics in Space III*, pages 1829–1832, Boston, MA, November 1992.

[Firby, 1989] R. James Firby. Adaptive execution in complex dynamic worlds. Technical Report YALEU/CSD/RR 672, Yale University, 1989.

[Gat, 1992] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. National Conference on Artificial Intelligence*, pages 809–815, San Jose, CA, July 1992.

[Georgeff and Lansky, 1987] Mike Georgeff and Amy Lansky. Reactive reasoning and planning. In *Proc. National Conference on Artificial Intelligence*, pages 972–978, Seattle, WA, July 1987.

[Hoffman and Krotkov, 1991] Regis Hoffman and Eric Krotkov. Perception of rugged terrain for a walking robot: True confessions and new directions. In *Proc. of the International Workshop on Intelligent Robots and Systems*, pages 1505–1510, Osaka, Japan, November 1991.

[Simmons *et al.*, 1990] Reid Simmons, Long Ji Lin, and Chris Fedor. Autonomous task control for mobile robots. In *Proc. IEEE Symposium on Intelligent Control*, Philadelphia, PA, September 1990.

[Simmons *et al.*, 1992] Reid Simmons, Eric Krotkov, William Whittaker, et al. Progress towards robotic exploration of extreme terrain. *Journal of Applied Intelligence*, 2:163–180, 1992.

[Simmons, 1992a] Reid Simmons. Concurrent planning and execution for autonomous robots. *IEEE Control Systems*, 12(1):46–50, February 1992.

[Simmons, 1992b] Reid Simmons. Monitoring and error recovery for autonomous walking. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, pages 1407–1412, July 1992.

[Singh and Simmons, 1992] Sanjiv Singh and Reid Simmons. Task planning for robotic excavation. In *Proc. IEEE Conference on Intelligent Robots and Systems*, Raleigh, NC, July 1992.

[Spivey, 1992] J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice Hall, second edition, 1992.

S35-63
207067

N94- 30562

6057

AIAA-94-1211-CP

p. 5

A Survey of NASA and Military Standards on Fault Tolerance and Reliability Applied to Robotics

Joseph R. Cavallaro and Ian D. Walker
cavallar@rice.edu ianw@rice.edu
Dept. of Electrical and Computer Engineering
Rice University, Houston, TX 77251

Abstract

There is currently increasing interest and activity in the area of reliability and fault tolerance for robotics. This paper discusses the application of Standards in robot reliability, and surveys the literature of relevant existing standards. A bibliography of relevant Military and NASA standards for reliability and fault tolerance is included.

1 Introduction

Applications of intelligent robots are expanding to remote and hazardous environments, such as nuclear waste handling, and undersea and space operations. Fault tolerance and reliability are of paramount importance in these environments, since repair is often difficult, and failures potentially catastrophic.

However, efforts in robot reliability and fault tolerance have often been piecemeal and application-specific. The formality and consistency across applications of Standards and Protocols are successfully applied to many other engineering areas.

The Standards documentation spans several different categories. There are Handbooks (Reliability of Electronic Equipment [7], MIL-HDBK-217F, Fault Tree Handbook [25], NUREG-0492), Parts Specifications and Standards (Aircraft Data Bus [13], MIL-STD-1553B, Aircraft 28V DC Motors [10], MIL-M-8609B) Procedures and Programs (Failure Modes, Effects Analysis [14], MIL-STD-1629A, System Safety Program [20], MIL-STD-882), and Data Item Descriptions (Format for reports required under procedures FMEA [2], for example DI-R-7085A).

Standards utilization varies widely (Reliability Data in MIL-HDBK-217F covers a variety of components under thermal stress, some Standards include handbooks on failure data for electronic equipment, an Aircraft Survivability Program Standard [16], MIL-STD-2072, references documents from the Defense Nuclear Agency on Nuclear Weapon Effects on Aircraft). However, most Standards deal with non-nuclear environments, and further studies are needed for hazardous waste sites. There are also Standards for Software Quality [3], for example DOD-STD-2168.

This paper will discuss the potential application and tailoring for robotics applications of the existing standards, including the Robotic Industries Association (RIA) and American National Standard for Industrial Robots and Robot Systems standards. A standard has been developed for safety requirements [28], ANSI/RIA R15.06-1986 and a new standard is proposed for reliability [27], BSR/RIA R15.05-3-199X. For example, procedures for a failure modes and effects analysis (FMEA) described in standard MIL-STD-1629A, together with DI-R-7085A, allow tailoring of the specifications to the robot needs. We will note the use of FMEA in robot system reliability [1], together with ongoing work in architectures for robot fault detection and fault tolerance [30].

2 Standards Categories

The military standards literature can be divided into a number of major categories [26, 31]. These include handbooks and parts specifications useful in the characterization of components for a system. Other documents describe procedures and programs which are useful for design, analysis, or system operation. Additionally, data item description documents provide standardized report generation procedures which are useful for system specification and procurement.

2.1 Handbooks

One of the more widely used military standards handbooks is MIL-HDBK-217F, Reliability of Electronic Equipment [7]. This handbook provides tables to calculate failure rates for a number of electronic components from resistors and capacitors, to switches and relays, to motors and resolvers. Reliability data for mundane components, such as connectors, is presented along with failure estimates for complex integrated circuits, such as microprocessors. The failure rates are also based on the environment in which the component is expected to be used from benign ground use to extreme missile or cannon launch. Thermal effects on component reliability are considered very important in the derating analysis.

NASA has published a standard for reliability [24], NASA-TM-4322 which references the data in MIL-HDBK-217F. In the NASA document, tables are given which further derate components for space use beyond the factors given in MIL-HDBK-217F. Examples of failure rate calculations are given in section 3.

The use of MIL-HDBK-217F is described in a tutorial handbook, MIL-HDBK-338-1A, Electronic Reliability Design Handbook [8]. A valuable handbook for system reliability analysis is published by the Nuclear Regulatory Commission as NUREG-0492, the Fault Tree Handbook [25].

2.2 Parts Specifications

In addition to the more generic handbooks, there is a large collection of standards for individual parts. Many of the standards were developed for a particular military project which required a specific design. Many of the standards for aircraft components may be useful for specifying the reliability of robotic assemblies. Electric motors [10] are described in MIL-M-8609B while hydraulic actuators are described in MIL-A-5503E [5] and MIL-M-7997C [9]. The bibliography lists other standards for components such as shaft encoders and various switches which could be used as limit switches. As an example, the standard for an aircraft computer data bus, MIL-STD-1553B [13] was used in the design specification of the NASA Flight Telerobotic Servicer (FTS) project [22].

2.3 Procedures and Programs

When a particular system is in the design phase, it is useful to perform a failure modes and effects analysis. Tools such as fault trees may be used to generate this analysis. In addition, the analysis needs to be customized for the system and its intended use. In MIL-STD-1629A, a procedure for a generic Failure Modes and Effects Analysis [14] is given. For systems that may cause harm to people or other equipment, a safety protocol should be developed. In MIL-STD-882, a System Safety Program [20] which identifies hazards is described.

2.4 Data Item Descriptions

Data item descriptions describe the format for reports required under various procedures. For example, reports generated for a failure modes and effects analysis of a system would be written in a format given [2] by DI-R-7085A. NASA has similar doucmentation formats such as the

NASA Assurance Specification Documentation Standard [23], NASA-TM-101859. These format specifications are valuable in generating design, operation and maintenance documents.

3 Failure Probability

As detailed in [1, 25], the probability of a component failure can be calculated from a failure rate for the component [4]. Given a constant failure rate λ and using the exponential distribution, the probability of failure at time t is [1]:

$$p(t) = 1 - e^{-\lambda t},$$

the reliability of the component in the system is given by

$$R(t) = 1 - p(t) = e^{-\lambda t},$$

and the mean time to failure (MTTF) is given as

$$MTTF = 1/\lambda.$$

If the failure rate is small, the probability of failure is often approximated as λt [25]. An expert system can be used to model component decay by using time-dependent probabilities [25]. A small update routine monitors the system time and modifies the basic probability facts during the life of the robot.

Various methods can be used to determine the failure rate λ. For example, in [7], the average failure rate λ_m for a D.C. motor is estimated as

$$\lambda_m = [(t^2/\alpha_B^3) + (1/\alpha_W)]$$

failures per 10^6 hours, where t is the operating time period for which λ_m is the average failure rate, α_B is the bearing characteristic life, and α_W is the winding characteristic life of the device. Both α_B and α_W depend on the ambient temperature for the device, with expressions given in [7]. For an ambient temperature of $20^o C$, an operating period of 100 hours, the data in [7] gives a failure rate of 6.3×10^{-7} failures per hour.

Also in [7], the average failure rate λ_r for a resolver is given as

$$\lambda_r = \lambda_b \pi_S \pi_N \pi_E$$

failures per 10^6 hours, where λ_b is the base failure rate (exponentially related to ambient temperature), π_S is a factor related to the device size, π_N is related to the number of brushes, and π_E is an environmental factor. For a small resolver with 4 brushes and the same ambient temperature as the motor above in a (possibly mobile) ground-based environment, the failure rate λ_r is found from data in [7] to be 1.6×10^{-6} failures per hour.

The calculation of failure rates is useful to complete a fault tree analysis. Once failure rates have been found for the components, it is possible to compute failure probabilities from this data. Within the fault trees, these failure probabilities are combined through the logic gates using simple multiplication and addition [25]. The probability of failure for the output event of an AND-gate is the product of all the input probabilities and a conservative estimate of the output event probability for an OR-gate is the sum of the input probabilities.

In [29], an expert system is used to maintain the probability of failure for each node within the fault tree. The operator initializes only the basic components (leaves) in the tree with appropriate probability facts. The expert system then initializes the probabilities for inner nodes of the tree by combining the basic component probabilities through the gates in the tree structure. For purposes of design and planning, it is possible to explore the effects of individual component reliability on the overall reliability of the system.

4 Conclusions

Fault tolerance is of increasing concern in the design and use of robots. The military, nuclear power, and space programs have developed a number of reliability standards for the design and analysis of complex systems. The application of these standards to the design of robots will be extremely important in many applications, particularly in hazardous environments. Industrial groups, such as RIA, have proposed standards for safety and are currently developing standards for reliability.

Acknowledgments

This work was supported in part by NSF under grants DDM-9202639 and MSS-9024391 and by DOE Sandia National Laboratory Contract #18-4379A.

References

[1] B.S. Dhillon. *Robot Reliability and Safety.* Springer-Verlag, New York, NY, 1991.

[2] DI-R-7085A. Failure Mode, Effect, and Criticality Analysis Report. Data item description, DOD, September 1984.

[3] DOD-STD-2168. Defense System Software Quality Program. Technical report, DOD, ARDEC, Picatinny Arsenal, NJ, April 1986.

[4] V. H. Guthrie and D. K. Whittle. RAM Analysis Software for Optimization of Servomanipulator Designs. DOE SMALL BUSINESS INNOVATIVE RESEARCH (SBIR) PROGRAM REPORT JBFA-101-89, JBF Associates, Inc., Knoxville, TN, March 1989. Performed for Oak Ridge National Laboratory.

[5] MIL-A-5503E. Actuators: Aeronautical Linear Utility, Hydraulic, General Specification for. Military specification, DOD, ASD, Wright-Patterson AFB, OH, January 1986.

[6] MIL-E-85082(AS). Encoders, Shaft Angle to Digital, General Specification for. Technical report, DOD-Naval Air Systems, Washington, DC, September 1977.

[7] MIL-HDBK-217F. Reliability Prediction of Electronic Equipment. Technical report, DOD, Rome Laboratory, Griffiss AFB, NY, January 1990.

[8] MIL-HDBK-338-1A. Electronic Reliability Design Handbook. Technical report, DOD, Rome Laboratory, Griffiss AFB, NY, October 1988.

[9] MIL-M-7997C. Motors, Aircraft Hydraulic, Constant Displacement General Specification for. Military specification, DOD, NAEC, Lakehurst, NJ, September 1981.

[10] MIL-M-8609B. Motors, Direct-Current, 28 Volt System, Aircraft General Specification for. Military standard, DOD, December 1987. Notice 1.

[11] MIL-S-8805/57B. Military Specification Sheet Switch, Actuator, Plunger Type. Technical report, DOD, June 1990. Notice 1.

[12] MIL-S-8805/59A. Military Specification Sheet: Switch, Actuator, Roller Leaf. Technical report, DOD, June 1990. Notice 1.

[13] MIL-STD-1553B. Aircraft Internal Time Division Command/Response Multiplex Data Bus. Technical report, DOD, ASD, Wright-Patterson AFB, OH, September 1978.

[14] MIL-STD-1629A. Procedures for Performing a Failure Mode, Effects and Criticality Analysis. Technical report, DOD, NAEC, Lakehurst, NJ, November 1980.

[15] MIL-STD-2069. Requirements for Aircraft Nonnuclear Survivability Program. Technical report, DOD, NAEC, Lakehurst, NJ, August 1981.

[16] MIL-STD-2072. Survivability, Aircraft; Establishment and Conduct of Programs for. Technical report, DOD, Dept. of Navy, Air Systems Command, August 1977.

[17] MIL-STD-781D. Reliability Testing for Engineering Development, Qualification, and Production. Technical report, DOD, October 1986.

[18] MIL-STD-785B. Reliability Program for Systems and Equipment Development and Production. Technical report, DOD, ASD, Wright-Patterson AFB, OH, September 1980.

[19] MIL-STD-810E. Environmental Test Methods and Engineering Guidelines. Technical report, DOD, July 1989.

[20] MIL-STD-882B. System Safety Program Requirements. Technical report, DOD, AFSC, Andrews AFB, Washington, DC, March 1984.

[21] MIL-T-48460B(AR). Military Specification: Tachometer, Rate, Computer: 11732700. Technical report, DOD-Army, ARRADCOM, Dover, NJ, January 1987.

[22] NASA. Computer Based Control System Noncompliance Report for Computer Independent Hazard Control System. REPORT, NASA Goddard Flight Center, Greenbelt, MD, September 1991.

[23] NASA-TM-101859. Assurance Specification Documentation Standard and Data Item Description: Volume of the Information System Life-Cycle and Documentation Standards. Technical Report Technical Memorandum 101859, NASA Office of Safety, Reliability, Maintainability, and Quality Assurance, Washington, DC, February 1989.

[24] NASA-TM-4322. NASA Reliability Preferred Practices for Design and Test. Technical Report Technical Memorandum 4322, NASA Office of Safety and Mission Quality, Washington, DC, April 1991.

[25] NUREG-0492. Fault Tree Handbook. Technical report, Nuclear Regulatory Commission, 1981.

[26] M. Pecht and E. Hakim. The Future of Military Standards: A Focus on Electronics. *IEEE Aerospace and Electronic Systems Magazine*, 7(7):16–19, July 1992.

[27] BSR/RIA R15.05-3-199X. Proposed American National Standard for Industrial Robots and Robot Systems - Guidelines for Reliability Acceptance Testing. Technical report, ANSI/RIA, 1993.

[28] ANSI/RIA R15.06-1986. American National Standard for Industrial Robots and Robot Systems - Safety Requirements. Technical report, ANSI/RIA, 1986.

[29] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Expert System Framework of Fault Detection and Fault Tolerance for Robots. In *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications. Proceedings of the Fourth International Symposium on Robotics and Manufacturing*, pages 793–800, Santa Fe, NM, November 1992. ASME Press, New York, NY.

[30] I.D. Walker and J.R. Cavallaro. Dynamic Fault Reconfigurable Intelligent Control Architectures for Robotics. In *Proceedings 1993 Fifth American Nuclear Society Meeting on Robotics and Remote Handling*, pages 305–312, Knoxville, TN, 1993.

[31] G. F. Watson. MIL Reliability: a New Approach. *IEEE Spectrum*, 29(8):46–49, August 1992.

N94- 30563

6058

AIAA-94-1212-CP

A PERFORMANCE ANALYSIS METHOD FOR DISTRIBUTED REAL-TIME ROBOTIC SYSTEMS: A CASE STUDY OF REMOTE TELEOPERATION

D.R. Lefebvre and A.C. Sanderson

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute, Troy, New York

[illegible]

Abstract

Robot coordination and control systems for remote teleoperation applications are by necessity implemented on distributed computers. Modeling and performance analysis of these distributed robotic systems is difficult, but important for economic system design. Performance analysis methods originally developed for conventional distributed computer systems are often unsatisfactory for evaluating real-time systems. The paper introduces a formal model of distributed robotic control systems; and a performance analysis method, based on scheduling theory, which can handle concurrent hard-real-time response specifications. Use of the method is illustrated by a case study of remote teleoperation which assesses the effect of communication delays and the allocation of robot control functions on control system hardware requirements.

1 Introduction

As ambitious robotic applications are envisioned and more complex robot designs attempted, the need increases for efficient methods to evaluate their performance. Many of these new applications will be implemented on distributed computers. For instance, remotely operated and multiple-robot applications are by their nature spatially distributed, and so necessitate a distributed real-time system for robot coordination and control. The introduction of multiple processors, communication delays, and probabilistic performance of common-access communication channels in distributed systems complicates prediction of their real-time performance.

Performance analysis methods for conventional distributed systems employ one of three approaches: simulation, stochastic models, or semantic models [7]. These methods have complementary strengths and weaknesses; so, several methods may be needed to analyze all aspects of system performance. The characteristics of these methods relevant to analyzing distributed real-time systems are summarized in Figure 1.

Simulation is arguably the most widely used approach. In a simulation model, the actual operation of the system is duplicated in software at an abstract level of detail. The fidelity of the simulation depends upon accurately representing the structural properties of the system such as precedence of operations and contention for resources; and its timing properties such as execution times, communication latencies, and sensor polling delays. A simulation can produce a full probability distribution of system response times; and so, provide a complete characterization of soft- and hard-real-time performance. Thorough characterization comes at a price: a high level of detail is needed for good accuracy, but is computationally expensive. Also, complex systems have an extremely large number of states that may necessitate excessively lengthy simulation duration to ensure that all states are exercised. For this reason, simulations are poor for proving system correctness and global properties such as boundedness and freedom from deadlock.

Stochastic models (e.g. Markov chains, queuing networks, Petri nets) are also commonly used for performance analysis, particularly for evaluating communication networks. In this approach, the system is idealized as a finite set of discrete states with known probability distributions for the transition rates between states. The model may be solved to estimate probability of each state as a function of time from which average system performance may be derived. For simple systems an efficient, analytical solution is often possible, and correctness and global properties may be determined. However, stochastic models of complex systems can be analytically intractable; requiring approximation methods which may compromise fidelity and increase computation.

[1]Copyright ©1994 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Semantic modeling is a less common approach to assess system performance that arises from computational science theory of program correctness. In this approach, the logical and temporal relationship between operations of the system are defined by process algebras or assertional calculi. Correctness and timeliness properties are then established by solving the model via a theorem prover. Semantic models are effective for proving that timing specifications are satisfied, but do not necessarily provide quantitative measures of system performance. The downfall of semantic models is their computational complexity; verification is impractical for large systems.

None of the three approaches described above is fully satisfactory for estimating performance of distributed systems having hard-real-time response requirements. In a hard-real-time system, response times must never exceed specifications; and so, the system must be analyzed for worst-case performance. Simulation can produce estimates of worst-case performance, but at a high computational cost which becomes prohibitive when the system is designed for multiple concurrent responses. Stochastic models give average response times only, and thus do not provide the information necessary to gauge performance of a hard-real-time system. Semantic models excel at proving correctness and global properties, but are poor at quantifying response times. A fourth approach, based on scheduling theory, is proposed in this paper to specifically address distributed hard-real-time systems.

In the new performance analysis method, a formal model describes distributed real-time system organization and its responses to external inputs. System software is modeled as independent tasks that communicate by messages. Application of scheduling theory enables the calculation of guaranteed response times for task executions and message deliveries. The model provides a framework for formulating a constraint satisfaction problem on processor and communication channel schedules and on real-time requirements whose solution defines system response times. The performance analysis problem may be solved to minimize weighted system response time or to minimize hardware cost while meeting response time requirements. The subsequent paper sections outline the system model, show the formulation of the constraint satisfaction problem, and then illustrate its use in an example.

While this work has been motivated by the design of robot coordination and control systems, the performance analysis techniques are believed to have broader application to many distributed real-time systems.

Method	Provably Correct	Real-Time Response		Computing Expense	Limitations
		Soft	Hard		
Simulation	Poor	Good	Good	High	high level of detail needed for fidelity
Stochastic Models	Fair	Good	Fair	Moderate	some problems intractable; approx. may lead to poor fidelity
Semantic Models	Excel.	Poor	Fair	High	v. difficult to develop model; large problems intractable
Scheduling Theory	(Good)	Fair	Good	Low	pessimistic for soft-real-time;

Figure 1: Performance Analysis Method Comparison

2 Performance Analysis System Model

Distributed computer systems are composed from multiple, independent processors connected by communication links. The characteristics of distributed systems can vary widely as the result of bandwidth and propagation delay of the interprocessor connections. At the extremes are "tightly-coupled" *multiprocessor* computers in which processors share a high-speed bus, and "loosely-coupled" *multicomputer* systems which comprise separate computers connected by a network. Processor independence distinguishes distributed systems from parallel computers in which processors typically are identical, and share data streams and/or instruction decoding.

The proposed formal model can represent distributed systems with arbitrary communication network topography; and so, can model the full range from multiprocessor to multicomputer system. In fact, in the model, a single node of a multicomputer network may be a complete multiprocessor. The new method is particularly useful for loosely-coupled systems, where access to communication channels as well as processor usage must be scheduled, since few analysis techniques are available for this class of distributed system.

Because the independent computers of a distributed systems do not share physical memory, any data to be exchanged between processors must be transmitted across a communication link. The most common way to design distributed software that accommodates this restriction is to organize functions as independently-executing tasks that communicate via messages. This paradigm of tasks and messages is used in the formal model to define the system software, although the definition of a message has been generalized to include less-structured signals such as sensor inputs or control outputs.

Some tasks must execute on specific processors; for instance, a sensor polling task must run on the processor that is interfaced to the sensor hardware. However, in general, there are many choices of how to distribute software on the hardware. The actual assignment of tasks to processors has a strong influence on system performance; and so, must be specified for performance to be predicted. Optimal task assignment is an important design problem for distributed systems. We are currently experimenting with use of the new performance analysis method to guide task assignment [4].

Robotic systems, and indeed most real-time systems, interact with their environment. Sensors gather data to characterize the environment. The control system monitors sensors to detect occurrence of specific conditions or events to which the system is designed to respond. And the system effects changes to the environment through its actuators; thus forming a closed-loop system. Also, in most systems, a human operator may intervene to modify goals or to initiate actions. Performance of robotic systems may be measured in many ways: accuracy, reliability, cost, etc. As we are primarily concerned with the computer system providing robot coordination and control, performance will be defined as the end-to-end response time from when a condition can be sensed until a control signal is sent to actuators. Therefore, system response requirements are identified by input-output events and a response time specification. The requirement specifies a maximum response time since we are dealing with ***hard*** real-time systems.

From this description we see that four components are needed to fully describe a distributed real-time system:

- software system model
- hardware system model
- assignment of tasks to processors
- system response requirements

In the definition of each model component, covered in the following subsections, we have attempted to describe distributed real-time systems in terms that are

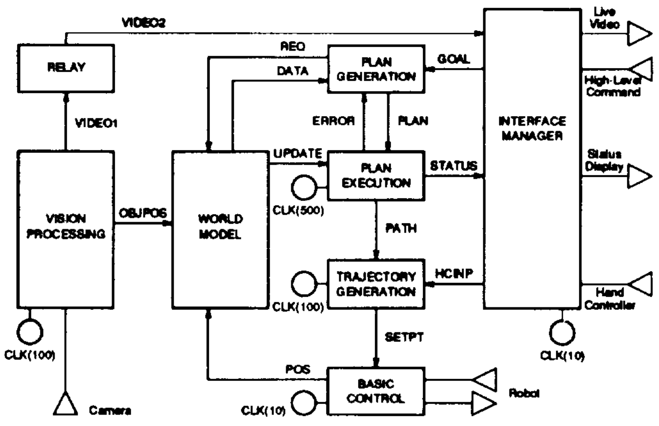


Figure 2: Software Model of Teleoperation Example

as similar as possible to how they are actually constructed. While this tends to specialize the model, it has the benefit of providing a more natural representation of a system implementation which, hopefully, improves ease of use and accuracy.

2.1 Software System Model

A distributed robotic application is typically constructed from many, concurrent tasks that execute on multiple processors, and communicate by message passing between tasks, or between task and sensor or actuator. Each task corresponds to a software module, and the resulting system may be described by a directed graph with nodes corresponding to tasks and arcs representing messages. Each task is a separate software module that may execute periodically or upon demand ("aperiodic" or "event-driven"). This *system level* graph defines the topography of the communications between tasks.

Figure 2 shows a system level view of a simplified control system for the teleoperated robot example that will be described in Section 4. The example employs a hierarchical architecture loosely modeled on the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) [1]. System software is modeled with five periodic tasks and three event-driven tasks, which are shown as boxes in the figure. Periodic tasks are identified by their clock inputs (circles). Input (sensors, keyboards) and output (actuators, displays) devices are represented by triangles whose orientation denote direction of data flow. Messages are shown as arrows from sending task to receiving task. A total of fourteen messages are transmitted between tasks in the example.

At a more detailed *module level*, each task is viewed as a finite state machine where task states or actions are nodes, and transitions between actions are directed arcs. The transition arcs are labeled with Real-Time Logic predicates [3] which define the condition causing the transition to occur. The purpose of the module level view is to define the response of an individual task to the input messages it receives. The finite state machine representation of the task allows a different computation time and different set of output messages to be defined for each input message.

Each action node in the finite state machine represents a deterministic sequence of operations that are delimited by a decision branch or a message transmission/arrival. When a node is entered it executes for a fixed time interval and then optionally sends a message prior to blocking in that state or transitioning to another. Computation times are associated with actions, while transitions are instantaneous. The optional messages produced by module actions correspond to the messages output from modules at the system level. Messages are identified only by type and bit length; the data values contained in a message are not considered.

Figure 3 shows the finite state machines for two tasks from the teleoperated robot example. The VISION PROCESSING task periodically acquires a camera image frame, transmits the frame as a VIDEO1 message, processes the image to locate objects in the robot's environment, and then outputs the positions of the objects in a OBJPOS message. Task processing is initiated when a CLK signal is received; and, when complete, the task returns to Idle Wait state to await the next signal. Figure 3b shows the finite state machine for the aperiodic PLAN GENERATION task. This task is invoked by the arrival of a message rather that a clock signal; and contains two paths so that GOAL and ERROR messages may be processed differently. Note that execution of the task is interrupted at Action 4 while it waits for requested data. Definition of periodic and aperiodic tasks are essentially the same at the module level — differing only by whether a clock signal or a message activates the task.

2.2 Hardware System Model

The purpose of the hardware system model is to describe how processors are interconnected, and the capabilities of processors and communication channels. The principal capabilities of interest are processor

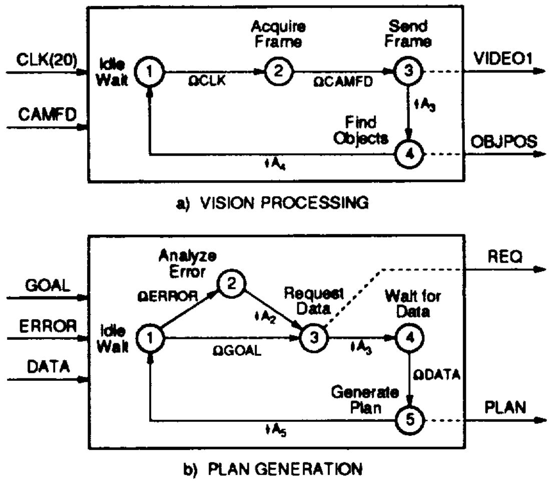


Figure 3: Finite State Machines for Example Modules

speeds, and communications bandwidth and propagation delay.

Processor interconnections are represented by a *hardware graph* in which graph nodes correspond to processors, and graph arcs indicate the one-hop communication links between processors. Dedicated, unidirectional communication channels are shown as directed arcs; and shared communication channels (half-duplex or broadcast media) as sets of non-directed arcs. Any connection topography can be modeled in this way including loosely-coupled multicomputer networks, bus-based multiprocessors, and combinations of the two.

Figure 4a is a diagram of a multicomputer system with four single-processor workstations and a 4-processor multiprocessor connected by a local area network. One sensor and one actuator are interfaced to the multiprocessor. Figure 4b is the corresponding hardware graph. Note how the shared multicomputer LAN and the shared multiprocessor bus are expanded into sets of bi-directional links that fully interconnect all processors sharing each communication medium. There are no dedicated links in this example.

2.3 Task Assignment

The distribution of software modules onto computer hardware is described by first numbering all tasks and processors, and then defining an *assignment function* α which maps a task to a processor. Thus if task t_i is assigned to processor p_j then $\alpha(i) = j$. This definition allows us to reference the hardware properties of the processor on which a task runs.

A similar assignment function can be defined to ref-

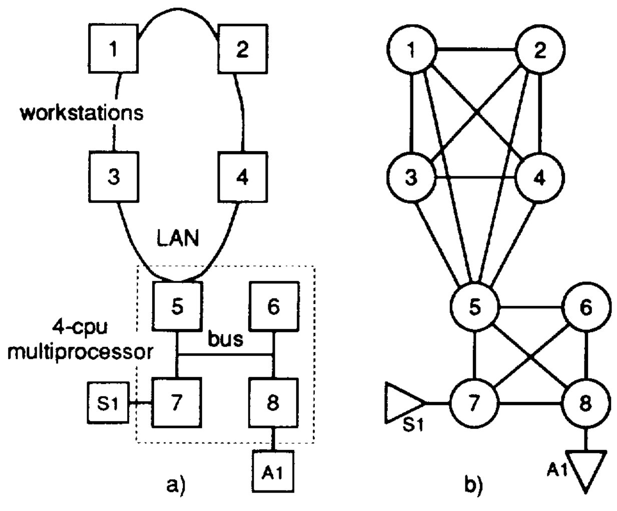


Figure 4: Hardware System Model Example

erence the communication hardware used by a message; thus if message m_i is assigned to communication link l_j then $\alpha(i) = j$. Once tasks are assigned to processors the communication link over which a message travels is defined. Therefore the message assignment function can be derived from the task assignment function plus the software and hardware system models; and so, we only need to specify the task assignment function.

2.4 Response Requirements

For this work, the principal performance measure is response time. System response time is defined as the time interval between occurrence of an external event and the system response. When sensor polling delays and actuator response times are factored out, the response time can be expressed as the time between an external input (sensor reading, operator command, etc.) and an external output (control signal, display update, etc.) of the control system.

System response requirements specify the events to which the system must respond, the expected actions, and response time. Requirements correspond to the environmental constraints on the robotic system. We will consider only hard-real-time requirements in which a maximum response time is specified and must be satisfied.

Most robotic systems will respond to many different events; and so, multiple response requirements will be defined. In hard-real-time applications, the system is expected to process concurrent events within their maximum response times under all load conditions. It is this requirement for concurrent responses that makes analysis of hard-real-time systems difficult. Contention for processors and communication channels will vary as the mix of concurrent events and their relative overlap varies. For instance, it is difficult to ensure that sufficient simulations have been performed such that the worst-case combination of concurrent events is modeled. And, average response times obtained from stochastic models provide no information regarding response degradation under load. A key advantage of a scheduling theory-based approach is that its results hold for all phasings of task invocations, i.e. degree of overlap of concurrent events.

3 Formulation of Performance Analysis Constraint Satisfaction Problem

With the information contained in the system model described above, a constraint satisfaction problem can be formulated whose solution yields an estimate of system response times. Performance of the distributed robotic system is defined by a set of constraint equations relating hardware, software, and response times. These equations are presented in the following subsections.

This system of equations is underconstrained; and so, a cost function is added to reduce the degrees of freedom. Different solution objectives can be achieved with different cost functions. In particular, the constraint equations may be solved to yield minimum system response times for fixed hardware capacities, or to find minimum-cost hardware which can meet all system response time requirements.

The problem is summarized as:

- Minimize system response times <u>or</u> hardware cost
- Subject to:
  1. Having a feasible schedule on every processor and communication channel
  2. Meeting system response time requirements
  3. Satisfying bounds on individual task and message response times

3.1 Cost Functions

If no constraints are mutually exclusive then the constraint satisfaction problem can be solved. However, since it is typically underconstrained, the problem can have an infinite number of solutions. A cost function is included which introduces additional constraints to ensure that only one solution is produced.

Through our choice of cost function we can direct the solution to achieve various design objectives.

System response time is one possible cost function. Since the system may have multiple responses, a weighted sum of response times is used to give a single-valued function. This allows us to emphasize one system response over another. A large penalty is assigned for exceeding a system response requirement, so all requirements are met before responses are further minimized. When this cost function is used, hardware capacities are held constant; hence, this form is useful for evaluating existing hardware.

As an alternative, hardware capacities may be allowed to vary, and hardware cost used as the cost function. The problem solution yields values for hardware capacities as well as system response times. This form of the constraint satisfaction problem is useful for evaluating proposed hardware designs. The example performance analysis in Section 4 is formulated in this manner.

3.2 Scheduling Constraints

A principal distinction between performance analysis methods is how they handles resource contention. Analysis methods for real-time systems must be able to represent the order of internal system events so that resource contention can be modeled. Usually this means that the protocols for scheduling task executions and message deliveries must be known. Simulation methods use this information directly; while stochastic models represent resource contention probabilistically. The scheduling-based performance analysis method presented here requires that a priority-based, preemptive scheduling protocol be employed for both processors and communication channels. Real-time operating systems typically implement such protocols for processors; however, communication protocols supporting time-constrained messages are recent developments [9][2], and are much less common.

The reason the scheduling-based method is restricted to priority-based, preemptive protocols is that it depends on their predictable properties. With this class of scheduling protocol the execution time of the highest priority task is always known, and the worst-case execution times of lower priority tasks can be determined by assuming all higher priority tasks must execute first. In 1973, Liu and Layland [6] proved several properties of priority-based, preemptive scheduling protocols and introduced an analysis technique known as the *rate monotonic scheduling algorithm*. They established criteria for multiple tasks executing periodically on a single processor that, when satisfied, guarantee a schedule can be found in which all tasks meet their execution deadlines. They also showed that an optimal schedule is obtained by assigning priorities based on task periods — shortest period task has highest priority. The original work on scheduling uniprocessors has been extended to systems with aperiodic tasks and to shared communication channels, and is now referred to as generalized rate monotonic scheduling [5][8].

In the proposed performance analysis method, scheduling theory criteria are used to identify the conditions under which a set of tasks [messages] can be scheduled such that they are guaranteed to meet execution [delivery] deadlines. These deadlines are then used as guaranteed response times. We have developed a modified form of the generalized rate monotonic scheduling algorithm which applies to the robotic system model with event-driven tasks and real-time constraints.[2] This modified scheduling criterion gives the minimum speed S^* of a processor [or communication link] required to successfully schedule the tasks [or messages] assigned to it:

$$S_j^*(\bar{C},\bar{r},\bar{\theta}) = \max_{\{1\leq i\leq N_t\}} \min_{\{\tau\in SP_i\}} \left(\sum_{n=1}^{i} C_n \left\lceil \frac{\tau}{\theta_n} \right\rceil \right) / \tau \tag{1}$$

where $\bar{C}$, $\bar{r}$, and $\bar{\theta}$ are vectors of computation times, deadlines (guaranteed response times), and periods of tasks [messages] assigned to processor [link] j, respectively. N_t is the number of assigned tasks [messages], and SP_i is the set of critical scheduling points as defined by:

$$\begin{aligned} SP_i &= \{(k-1)\theta_j + r_j \mid {\scriptstyle j=1,\ldots,i;\, k=1,\ldots,\lfloor \frac{r_i+\theta_j-r_j}{\theta_j} \rfloor}\} \\ &\qquad \bigcup \{k \cdot r_j \mid {\scriptstyle j=1,\ldots,i;\, k=1,\ldots,\lfloor \frac{r_i}{\theta_j} \rfloor}\} \end{aligned}$$

Note that computation times C_i are normalized for a "standard" processor defined to have a *relative speed* of 1. Processor speed and S^* are expressed as relative speeds by ratioing to the standard processor. Messages and communication channels are treated in the same manner.

[2]Strictly speaking, since the technique uses deadlines rather than periods it should be referred to as deadline-monotonic scheduling. However, for clarity the more familiar term will be used.

The scheduling constraints require that the minimum relative speed S^* for a feasible schedule be less than or equal to the relative speed S of the processor or communication link:

$$S_j^*(\bar{C}, \bar{r}, \bar{\theta}) \leq S_j \quad \text{for every processor and link } j \quad (2)$$

The scheduling criterion defined by equation 1 essentially forms a ratio between demand for execution capacity (summation term) and available capacity (τ). The ratio is checked at critical scheduling points which occur at deadlines and periods. Execution capacity demand is calculated for all tasks of priority i and higher priority tasks which may preempt it. The minimum ratio over all scheduling points reflects the lowest speed at which these tasks are schedulable for a given priority. Finally, the ratio is checked for all priorities, and the worst case defines the relative speed needed to successfully schedule the assigned tasks or messages.

3.3 System Response Time Constraints

As defined in Section 2.4, system response requirements are specified in terms of the external event which invokes a response, the expected system action, and response time. An external event detected by the system's sensors will trigger a cascade of task executions and message transmissions. Many tasks may execute concurrently and multiple messages may contend for shared communication channels. The precedence of task executions and message transmissions associated with a particular event can be derived from the software model and is represented as a weighted directed acyclic graph called an *event response graph*. Graph arcs are weighted with task execution times and message delivery times, which are dependent on the underlying hardware capabilities. Since the graph is deterministic, a critical path through the graph can be found that defines system response time for the specific event.

As an example, consider the response of the system from Figure 2 to a high-level command input by an operator. The command is received by the INTERFACE MANAGER task which interprets the command and then transmits a GOAL message to PLAN GENERATION. In subsequent processing steps data is obtained from the WORLD MODEL, a plan created and sent to PLAN EXECUTION, and so on until the system response to the high-level command is produced at the robot. The complete sequence of task executions and message transmissions is shown in the event response graph in

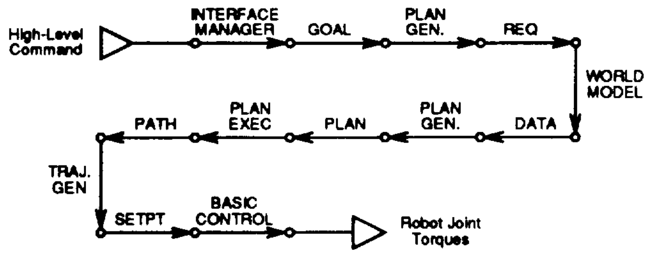


Figure 5: An Event Response Graph

Figure 5. This simple example has a linear critical path; but, in general, the critical path may contain parallel legs. Control system response time is calculated by summing guaranteed task execution times of the seven tasks in the graph including polling delays at the periodic tasks, plus guaranteed message delivery times of the six messages including propagation and switching delays, plus communication time associated with sensor input or actuator output. Note that the PLAN GENERATION task appears twice in the example event response graph. The first invocation of PLAN GENERATION is in response to a GOAL message, and the second in response to a DATA message. Execution times for PLAN GENERATION are different in each instance as defined in the module level finite state machine description of the task (see Section 2.1).

The fact that event response graphs must be deterministic does not prevent us from modeling probabilistic events such as failures. In these cases, an event response graph would be developed to represent the processing that occurs for each possible outcome; and, potentially, each outcome could have a separate hard-real-time response requirement. If a system is required to meet a response time specification even in the presence of failure then only the more restrictive situation would have to be modeled — probably the case including the additional processing to accommodate failure. Alternatively, a less demanding response time requirement could be defined for failure processing which would yield a less costly control system design. This type of analysis enables us to study tradeoffs between system reliability and cost.

An event response graph is constructed for each system response having a time requirement. Since guaranteed task execution times and guaranteed message delivery times are solution variables of the problem, system response time can be determined by summing the variables corresponding to the weights on the event response graph. The constraint equations are formed by requiring that system response time must be less than its requirement for each response:

$$\sum_{t_i, m_i \in CP_k} r_i \leq R_k \quad \text{for all responses k} \qquad (3)$$

where r_i is the guaranteed response time of task or message i, CP_k is the set of tasks and messages on the critical path for response k, and R_k is the system response time requirement.

3.4 Task/Message Response Time Bounds

Response time for an individual task or message is bounded. Response time can not be less than the time required to execute the task or transmit the message, and is not allowed to be greater than its period. This upper bound results from a restriction that at most one invocation of a task is allowed to execute at a time. For aperiodic tasks, a parameter analogous to period is specified to be the minimum interval between executions. These bounds place the following constraints on guaranteed response times:

$$\frac{C_i}{S_{\alpha(i)}} \leq r_i \leq \theta_i \quad \text{for all tasks and messages} \qquad (4)$$

where $S_{\alpha(i)}$ is the relative speed of the processor to which t_i or m_i is assigned (recall that α is the assignment function), θ_i is the period or minimum interarrival time of the task or message, and the other terms retain their earlier definitions.

Task/message response time bounds can be represented as simple variable bounds for constraint satisfaction problems with constant processor and communication channel speeds since all of the terms in the calculation of the lower and upper bounds would be known and constant. However, if hardware speeds are solution variables, then the lower bounds must be incorporated as nonlinear constraint equations.

3.5 Solving Constraint Equations

In summary, to analyze the performance of a distributed robotic system we first define the system by the model outlined in Section 2; then form the system of constraints from equations 2, 3, and 4. The constraint satisfaction problem is solved to minimize the cost function, i.e. to minimize weighted system response time, or to minimize hardware cost. The solution provides times for all system responses, guaranteed response times for task executions and message deliveries, and processor and communication channel speeds.

A nonlinear programming method is needed to solve the constraint equations. Unfortunately, although equation 1 is continuous it is not smooth. Therefore, nonlinear methods such as sequential quadratic programming and others that require smooth derivative information can not be used. The system of constraints has been successfully solved with a successive linear programming approach. We believe that this approach works because the partial derivatives of equation 1 are piecewise-linear.

4 Example Use of Analysis Method

This section presents an example use of the new performance analysis method for design of the control computer system of a teleoperated robot. The minimum-cost hardware formulation will be used to select capacities of processors and communication links for various design conditions of communication delay and task assignment.

Control software is organized in a "NASREM-like" architecture as seen earlier in Figure 2. The standard components of sensory processing, world modeling, task decomposition, and operator interface are all included; however, only the task decomposition functions are modeled in sufficient detail to show a hierarchical organization. The eight tasks comprising the system are listed in Table 1 with their relative computation times and execution periods. Note that the task decomposition functions of PLAN GENERATION, PLAN EXECUTION, TRAJECTORY GENERATION, and BASIC CONTROL form a hierarchy with execution period differing by an order of magnitude between levels. Parameters for the messages transmitted among the tasks are listed in Table 2.

Task	Comp Time, ms	Period, ms
Basic Control	4	10
Traj. Generation	30	100
Plan Execution	50	1000
Plan Generation	2000	-
World Model	50	-
Vision Processing	170	100
Video Relay	0.1	100
Interface Manager	10	10

Table 1: Task Parameters for Example

Figure 6 shows the control system hardware for the teleoperation example. It includes a *local* processor at ground station, a *remote* processor at an or-

Message	Length, kbits	Period, ms
GOAL	7.8	-
PLAN	7.8	-
PATH	3.7	1000
HCINP	0.4	10
SETPT	1.1	100
POS	0.4	10
OBJPOS	7.3	100
UPDATE	7.8	100
STATUS	7.8	1000
ERROR	0.1	-
REQ	0.8	-
DATA	7.8	-
VIDEO1	25	100
VIDEO2	25	100

Table 2: Message Parameters for Example

bital facility, and *control* and *vidpp* processors on the robot to support dedicated control and video preprocessing functions. Three communication channels connect these processors: unidirectional *uplink* and *dnlink* channels between ground and orbit, and a radio network, designated *rnet*, for communications between robot processors and the orbital facility. The nominal assignment of tasks to processors locates VISION PROCESSING on *vidpp*, BASIC CONTROL on *control*, INTERFACE MANAGER on *local*, and all remaining tasks on the *remote* processor.

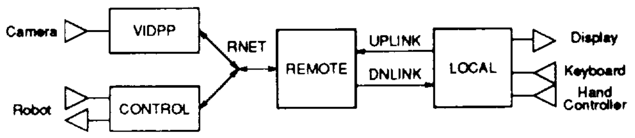


Figure 6: Hardware for Teleoperation Example

Five time-critical responses are specified, and serve as the hard-real-time system response time requirements. They are listed on Table 3. The control system must display information about the work site in three forms: live video at 10 frames/second, a reconstruction of the world model updated by object recognition, and a model showing robot position. The system must guarantee that data from each of the three sources is delivered to the INTERFACE MANAGER in 2.4 seconds (2400 ms) so that it can be fused into a consistent display. An operator controls the robot either indirectly through high-level commands, or directly via a hand controller. The system is expected to respond to high-level commands in 9600 ms, and hand controller input in 1200 ms. As covered in section 3.3, an event response graph is constructed for each system response requirement to identify the tasks and messages invoked to process each response.

Description	Mnemonic	Requirement
Display Live Video	LIVE_DSP	2400 ms
Display World Model	WM_DSP	2400 ms
Display Robot Position	ROB_DSP	2400 ms
Respond to HL Command	CMD_RSP	9600 ms
Respond to Hand Controller	HC_RSP	1200 ms

Table 3: System Response Requirements for Example

Relative costs for processors and communication channels were modeled with a power function: $cost = multiplier \times speed^{exponent}$. Ground facilities were assigned a 1.0 multiplier, orbital facilities were assumed to have an order of magnitude higher multiplier, and processors on the robot have an additional factor of two premium. Exponents of 2.0 for ground and 1.5 for orbital facilities were used. The cost model should have an additional additive factor; but the SLP solution method being used cannot support it.

4.1 Effect of Communication Delays

In the first design study, the effect of communication delay on processor and communication channel capacity is examined. The new performance analysis method is used to find the minimum hardware needed to guarantee that system response time requirements are achieved. Communication delays of 100, 500, and 1000 ms are studied. These values represent propagation and switching delays only; and so, may appear low compared to customarily quoted values which include scheduling delays due to traffic contention. The performance analysis method computes the scheduling delays.

Table 4 summarizes key results. As required, all system responses meet requirements. At 100 and 500 ms delays the high-level command response is limiting; whereas, the world model display response limits at 1000 ms delay.

Most non-limiting responses differ between cases by an amount equal to the difference in communication delay. This is a consequence of the solution method which focuses on the requirements that constrain hardware speed while essentially ignoring responses not at a limit. Requirements for non-limiting responses could be lowered to the reported values without affecting hardware speeds.

Faster processors and communication channels are needed as communication delay increases. At delays of

500 ms and lower, the more expensive *control* and *vidpp* processors are at minimum capacity needed to meet execution periods of their assigned tasks. A modest increase in *remote* processor speed is sufficient to accommodate a communication delay of 500 vs. 100 ms. However, for the 1000 ms delay, all processor speeds must be higher in order to meet system response requirements.

Total relative hardware cost differs little between 100 and 500 ms cases. However, the cost of the video preprocessor dominates total cost, thereby masking the 10% difference in cost of all other components between the cases. The design for 1000 ms delay is significantly more costly: 246% total and 137% system excluding *vidpp* costs relative to the 500 ms design.

The point of this analysis is not to draw conclusions regarding an admittedly over-simplified teleoperated robot application, but rather to demonstrate a possible use for the new performance analysis method. It is feasible to guide key design decisions, in this case by examining tradeoffs between control system costs and communication switching infrastructure, through use of real-time system analysis.

Case #	1	2	3
Comm Delays, ms	**100**	**500**	**1000**
System Responses, ms			
- LIVE_DSP	360	760	1210
- WM_DSP	1760	2060	**2400**
- ROB_DSP	1580	1880	2270
- CMD_RSP	**9600**	**9600**	9380
- HC_RSP	280	670	1160
Processor Capacity			
- control	0.40	0.40	0.65
- vidpp	1.70	1.70	3.55
- remote	1.22	1.36	1.52
- local	1.00	1.00	1.12
Comm Link Capacity			
- uplink	0.99	0.98	1.03
- dnlink	0.99	0.98	0.99
- rnet	0.10	0.11	0.11
Relative Cost			
- system ex vidpp	0.90	1.00	1.37
- vidpp	1.00	1.00	3.02
- total system	0.97	1.00	2.46

Table 4: Effect of Communication Delays

4.2 Effect of Task Assignment

Another use of the new performance analysis method is illustrated in this section as a design study evaluating the effect of task assignment on hardware cost. Communication delays are fixed at 500 ms for all cases. In the base case, PLAN GENERATION, PLAN EXECUTION, and TRAJECTORY GENERATION tasks execute on the *remote* processor, and the BASIC CONTROL task on the *control* processor.

The effect of moving first PLAN GENERATION and then PLAN EXECUTION to the *local* processor was modeled with the results shown in Table 5. Cost savings can be obtained by shifting computing load from expensive orbital processors to lower cost ground computers while still meeting response time requirements. For this simplified example, the analysis suggests that the savings may be substantial, and may motivate further study to assess the impact on other mission-critical factors such as the reliability and safety implications of remote computing.

Migrating dedicated processing at the robot to the somewhat less expensive computing available at Space Station may also be cost effective. Table 6 summarizes modeling results for moving the CONTROL task to the *remote* processor. This reassignment eliminates the *control* processor which is replaced by simpler hardware to receive control signals from *rnet*; and *remote* processor capacity is correspondingly increased. Communication latency of the control signals are on the order of 0.3–0.4 ms which should be acceptable. The full benefit of relocating control functions may not be achievable since some at-robot processing capability is required for safety functions which have not been modeled.

5 Future Work

Currently we are working to improving the efficiency of the performance analysis method; in particular, to increase robustness of the successive LP solution approach and to decrease computation time. The principal motivation for improving solution efficiency is so that the performance analysis method may be embedded in a genetic algorithm with the objective of finding near-optimal task assignments. If successful, this would provide a powerful tool for designing distributed real-time systems in which software module allocations and hardware are optimized concurrently.

Other activities are aimed at demonstrating the capabilities of the performance analysis method on a variety of robotic systems, and directly comparing results to those obtained from simulation and stochastic models. Theoretical and experimental verification of performance analysis tools will provide an important contribution to the field of robotics, and will form the

Case #	2	4	5
Comm Delays, ms	500	500	500
Task Assignments			
- Plan Gen.	remote	**local**	**local**
- Plan Exec.	remote	remote	**local**
- Traj. Gen	remote	remote	remote
- Control	control	control	control
System Responses, ms			
- LIVE_DSP	760	820	900
- WM_DSP	2060	**2400**	1970
- ROB_DSP	1880	2220	1790
- CMD_RSP	**9600**	**9600**	**9600**
- HC_RSP	670	670	680
Processor Capacity			
- control	0.40	0.40	0.40
- vidpp	1.70	1.70	1.70
- remote	1.36	0.92	0.81
- local	1.00	1.32	1.35
Comm Link Capacity			
- uplink	0.98	0.92	0.63
- dnlink	0.98	0.90	0.21
- rnet	0.11	0.11	0.08
Relative Cost			
- system ex vidpp	1.00	0.72	0.62

Table 5: Effect of Shifting Tasks to Local Proc

Case #	4	6	5	7
Comm Delays, ms	500	500	500	500
Task Assignments				
- Plan Gen.	local	local	local	local
- Plan Exec.	remote	remote	local	local
- Traj. Gen	remote	remote	remote	remote
- Control	control	**remote**	control	**remote**
System Responses, ms				
- LIVE_DSP	820	800	900	820
- WM_DSP	**2400**	2360	1970	2060
- ROB_DSP	2220	2170	1790	1870
- CMD_RSP	**9600**	**9600**	**9600**	**9600**
- HC_RSP	670	680	680	690
Processor Capacity				
- control	0.40	0	0.40	0
- vidpp	1.70	1.70	1.70	1.70
- remote	0.92	1.14	0.81	0.96
- local	1.32	1.32	1.35	1.36
Comm Link Capacity				
- uplink	0.92	0.92	0.63	0.84
- dnlink	0.90	0.93	0.20	0.60
- rnet	0.11	0.06	0.08	0.03
Relative Cost				
- system ex vidpp	0.72	0.66	0.62	0.51

Table 6: Effect of Shifting Control Task

basis for more efficient development of new robotics applications in the future.

Acknowledgment

This work has been conducted at the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) with support from NASA Grants NGT-10000 and NAGW-1333.

References

[1] J. S. Albus, H. G. McCain, and R. Lumia. NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical Report NIST Technical Note 1235, National Institute of Standards and Technology, Gaithersburg, MD, April 1989.

[2] B. Chen, G. Agrawal, and W. Zhao. Optimal synchronous capacity allocation for real-time communications with the timed token protocol. In *Proc. IEEE Real-Time Systems Symposium*, pages 198–207, Pheonix,AZ, 1992.

[3] F. Jahanian and A. K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Trans. on Software Engineering*, 12(9):890–904, September 1986.

[4] D.R. Lefebvre and A.C. Sanderson. Performance evaluation and task assignment in distributed robotic systems. In *Proc. IEEE Conf. on Robotics & Automation*, San Diego, May 1994 (submitted).

[5] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proc. IEEE Real-Time Systems Symposium*, pages 166–171, 1989.

[6] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of The Association for Computing Machinery*, 20(1):46–61, January 1973.

[7] A. A. Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press, 1986.

[8] L. Sha and S.S. Sathaye. A systematic approach to designing distributed real-time systems. *IEEE Computer*, 26(9):68–78, September 1993.

[9] W. Zhao, J. A. Stankovic, and K. Ramamritham. A window protocol for transmission of time constrained messages. *IEEE Trans. on Computers*, 39(9):1186–1203, September 1990.

538-63
207621
P. 4

6059

AIAA-94-1213-CP

PREDICTIVE SUFFICIENCY AND THE USE OF STORED INTERNAL STATE

N94-30564

David J. Musliner
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742
musliner@umiacs.umd.edu

Edmund H. Durfee and **Kang G. Shin**
Dept. of EE & Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122
{durfee,kgshin}@eecs.umich.edu

Abstract

In all embedded computing systems, some delay exists between sensing and acting. By choosing an action based on sensed data, a system is essentially predicting that there will be no significant changes in the world during this delay. However, the dynamic and uncertain nature of the real world can make these predictions incorrect, and thus a system may execute inappropriate actions. Making systems more reactive by decreasing the gap between sensing and action leaves less time for predictions to err, but still provides no principled assurance that they will be correct.

Using the concept of *predictive sufficiency* described in this paper, a system can prove that its predictions are valid, and that it will never execute inappropriate actions. In the context of our CIRCA system, we also show how predictive sufficiency allows a system to guarantee worst-case response times to changes in its environment. Using predictive sufficiency, CIRCA is able to build real-time reactive control plans which provide a sound basis for performance guarantees that are unavailable with other reactive systems.

Introduction

Traditional AI planning systems[3,10,15] have been criticized because they may spend large amounts of time building a plan that is out-of-date before it can be used, and thus the actions that the plan chooses may be inappropriate. For example, consider an intelligent autonomous vehicle that is waiting at a red light. When the light changes to green, the vehicle's sensors detect the change and, after some further processing, the system decides to move through the intersection and on to its destination. But, if the system spent too much time planning its entire route, the light may have changed back to red, and the plan's first action would be "inappropriate."

The work reported in this paper was supported in part by the National Science Foundation under Grants IRI-9209031 and IRI-9158473, and by a NSF Graduate Fellowship. The opinions, findings, and recommendations expressed in this publication are those of the authors, and do not necessarily reflect the views of the NSF.

In response to this critique, researchers have developed reactive systems[1,2,4,6,13] that perform little or no lookahead planning, instead choosing actions based on current sensor inputs. One goal of this behavior is to keep the selected actions appropriate to the current situation: because no planning is done, an action can be chosen quickly once sensor readings determine the current situation.

However, because computations can only occur at some finite speed, there will always be some delay between sensing and action. During this "sense/act gap," sensed information is stored in the system, either explicitly in memory modules or implicitly in the communication and processing mechanisms of the system. By choosing an action based on that stored information, the system makes an implicit *prediction* that the stored information will continue to provide a sufficiently accurate representation of the world.[5]

Because real-world systems are dynamic and somewhat uncertain, such predictions are inherently risky. Gat[5] suggested that these predictions and the associated stored internal state are useful only at higher levels of abstraction. We argue that, because the gap between sensing and action is inevitable, it is not the abstraction level but the magnitude of this delay (and the requisite prediction) that is critical. Systems in dynamic worlds must be "real-time," in the sense that the utility of the system's computations depends not only on their result, but on when that result is produced.[14] To guarantee correct performance, an intelligent real-time system must ensure that the actions it chooses are appropriate for the actual current state of the world, not just the state of the world that was last sensed.

Rather than solving the real-time problem, reactive systems simply operate in a "coincidently real-time" manner[7]— they function as quickly as possible, in the hopes that the sense/act gap will be reduced so much that significant world changes cannot occur during the gap. In this paper, we present a more rigorous approach to dealing with the sense/act gap. Our approach consists of *proving* that significant world changes cannot cause a particular selected action to be inappropriate, by verifying that the predictions spanning the sense/act gap are valid.

In the next section, we lay the foundations for this proof by defining the "interval of predictive sufficiency," or the time during which an observation provides sufficient evidence to accurately predict the value of some proposition. In the following section, we illustrate how explicit reasoning about predictive sufficiency can be implemented, with examples from CIRCA, the Cooperative Intelligent Real-time Control Architecture.[8,9] We describe how CIRCA uses predictive sufficiency while building real-time reactive control plans, to guarantee that the system will never choose inappropriate actions or miss real-time reaction deadlines. This paper concludes with sections discussing the type of knowledge that is required for reasoning about predictive sufficiency, and pointing out future directions for this research.

Defining Predictive Sufficiency

To accurately describe the concept of predictive sufficiency, we must begin with some notation. We will use a simple temporally-qualified modal logic to describe the state of a control system's knowledge. The logical statement $K(p[t_i], t_j)$ indicates that the system knows, at time t_j, that the proposition p holds at time t_i. For convenience, we will also use statements of the form $K(p[t_\alpha, t_\beta], t_j)$, indicating that the system knows, at time t_j, that p holds continuously over the time interval from t_α to t_β.

A control system's operations can be generally expressed as the acquisition of a sensory observation, the logical deduction of what that observation means about the state of the world at the time the observation was made, the deduction of the predictions that the observation allows the system to make about the world following the observation, and the selection of an action based on that knowledge. In our notation, we have:

$$\begin{tikzcd}
O[t_i] \arrow[d, "\text{interpret}"] \\
\forall p \in P : K(p[t_i], t_j) \arrow[d, "\text{predict}"] \\
\forall q \in Q : K(q[t_{q\alpha}, t_{q\beta}], t_k) \arrow[d, "\text{select}"] \\
a[t_{a\alpha}, t_{a\beta}]
\end{tikzcd}$$

where $O[t_i]$ is a sensory observation made at time t_i, P is the set of propositions which can be inferred about the world at time t_i from the observation, and Q is the set of propositions that can be predicted over the respective intervals $[t_{q\alpha}, t_{q\beta}]$. These intervals are the "intervals of predictive sufficiency," during which the observation O is sufficient to predict the value of the propositions Q. The time t_j is the time by which the system has derived its knowledge of P, and t_k is the time by which the system knows Q. Following those deductions, the action a is chosen and executed during the time interval $[t_{a\alpha}, t_{a\beta}]$.

We first use the concept of predictive sufficiency to show how an action can be guaranteed to be appropriate when it is executed. The key to avoiding an inappropriate action is to ensure that the value of the propositions used to choose an action will remain unchanged long enough to keep the action appropriate. This can be achieved by making action choices based on propositions whose intervals of predictive sufficiency cover the time during which the action's preconditions are necessary. More formally, suppose the action a requires a set of propositions R to hold during the respective intervals $[t_{ra}, t_{rb}]$. If $R \subseteq Q$ and $\forall r \in R : (t_{r\alpha} \leq t_{ra}) \wedge (t_{r\beta} \geq t_{rb})$, then the intervals of predictive sufficiency that are supported by the observation O ensure that the required propositions will hold as necessary.

For example, in the stoplight scenario described earlier, the vehicle agent will at some point make an observation confirming the proposition "the light is green" (P). This proposition alone is not sufficient to justify crossing the intersection, because there is no guarantee that, at the time t_j when P is known, the light is *still* green. The knowledge resulting directly from interpreting sensor readings can only describe past states of the world. However, if the system knows some information about the domain's dynamic behavior, it can derive additional propositions that describe the current and future worlds. In this example, the system might know that the traffic signal will switch to yellow for at least five seconds before it turns red. So, although the system does not know if the light is still green, it can conclude that, for at least five seconds after the light was seen to be green, the light must be either green or yellow, and the intersection will be "safe" to cross (Q). If the agent is sure that the time it takes to infer these propositions from its observations and cross the intersection is less than five seconds, it can guarantee that it will never be in the intersection during a red light.

Thus the addition of domain modeling information has allowed the system to make explicit predictions about the future state of the world, based on stored sensor readings. Given further information about the agent's own performance, these predictions are then shown to be sufficient to justify certain actions. This example illustrates how predictive sufficiency can cover the sense/act gap, avoiding inappropriate actions.

Implementing Predictive Sufficiency

In this section, we provide a high-level description of CIRCA and show how the prototype implementation of the architecture explicitly reasons about predictive sufficiency and makes guarantees about its behavior. Note that we do not claim this implementation is ideal; it serves only as a useful testbed to demonstrate the concepts of predictive sufficiency. More details on CIRCA are available in related publications.[8,9]

Figure 1 illustrates the architecture, in which an AI subsystem (AIS) and Scheduler cooperate to strategically plan and schedule a set of reactive behaviors that will cope with a particular expected domain situation. The parallel real-time subsystem (RTS) is guaranteed to accurately execute the behavior schedules, comprised of simple situation-response rules. In this paper, we are focusing on how the prototype AIS explicitly reasons about the sense/act gap and predictive sufficiency while planning reactions. Note that this lookahead planning is performed while previously-planned reactions are already executing on the RTS, so the planning process can be viewed as "off-line." To show how CIRCA uses predictive sufficiency, we must first briefly describe the system's world modeling techniques, which it uses to reason about the behavior of the world and the actions that the system should take to achieve its goals.

In the prototype implementation, the world model takes the form of a directed graph in which nodes represent possible states of the world and arcs represent instantaneous transitions between states. The status of ongoing processes in the world is explicitly encoded into the representation of a state. Important changes in process status thus correspond to transitions between states. The model distinguishes three types of state changes: *action transitions*, performed deliberately by the system's reactions; *event transitions*, due to external world occurrences; and *temporal transitions*, due to the passage of time and ongoing processes. Timing information is associated with each transition, representing constraints on how long the world must remain in a state until the transition may occur. We now illustrate how this model is used by the AIS to explicitly reason about the sense/act gaps that will occur when planned behaviors are executing on the RTS, and how the system guarantees that those gaps will not lead to inappropriate actions.

Avoiding Inappropriate Actions

Figure 2 shows an example portion of the graph-based world model for the stoplight scenario described above. Within the state descriptions, the model shows that the stoplight can take on its three signal colors, Red, Yellow, and Green. In the Yellow and Green states, it is safe for the agent to cross ("Safe2X"), but not in the Red state. In this simple example, we have abstracted out all of the agent's own state except for the indication of whether it has crossed the intersection or not. The different states of the traffic signal are connected by temporal transitions (double arrows) indicating that, as time passes, the signal will transition to subsequent states. Each temporal transition is labeled with the minimum possible delay before the transition occurs, perhaps derived from the agent's previous experience with this traffic signal. For example, the transition between the Red and Green states indicates that the signal will stay red for at least 60 seconds before turning green.

When planning reactions to operate in this domain, CIRCA does not build an enumeration of possible world states and then plan actions; instead, it dynamically constructs the graph model and the plan of actions together in a single depth-first search process, essentially similar to a forward-chaining STRIPS planner.[10] This process operates on a stack of world model states, examining each state in turn and planning actions that achieve goals and preempt temporal transitions that lead to failure.

To begin the planning process, the initial states are pushed onto the state stack. Then, as long as the stack is not empty, the system pops a state off the stack and considers it the *current state*. The system simulates all of the event transitions and temporal transitions that apply to the current state, yielding either new states that have not been examined yet or states that have already been processed (i.e., states for which actions have already been planned). New states are pushed onto the state stack, while old states are simply updated with the information that they have a new source state. The system then chooses an action to take in the current state, as determined by a heuristic scoring function.

For example, if the system is told that the "red" state $\mathcal{A}$ is its initial condition, it will first consider the applicable event and temporal transitions, pushing the new "green" state $\mathcal{B}$ onto the stack. The system will then try to plan an action for state $\mathcal{A}$; since the state is not safe for crossing, the only applicable action is `no-op` (shown as a dashed line in

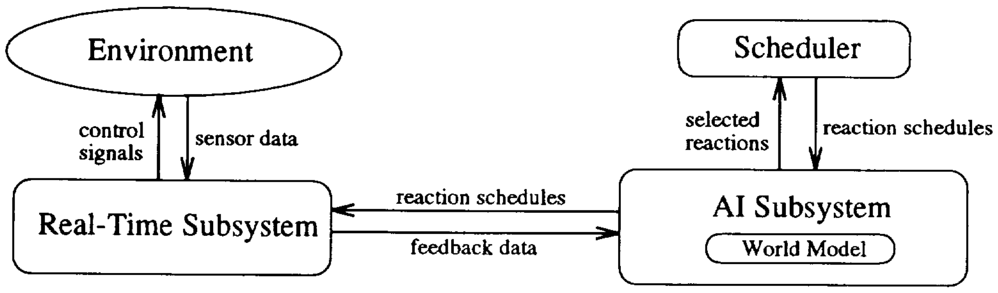


Figure 1: Overview of CIRCA.

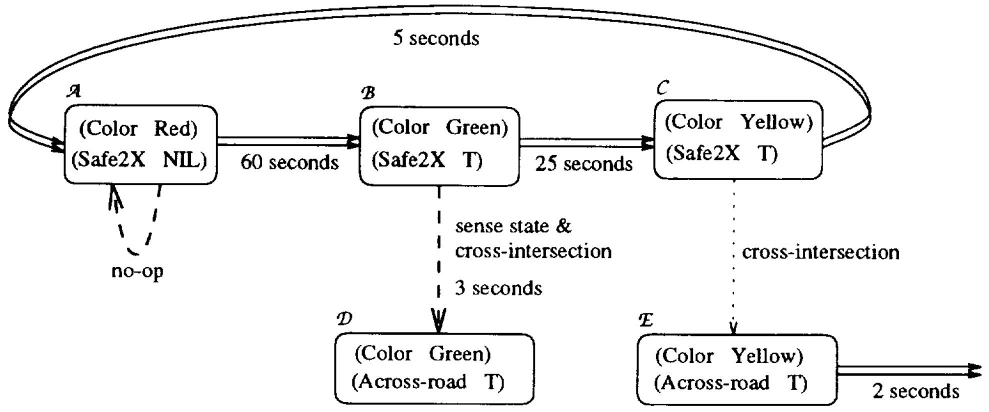


Figure 2: An abstracted portion of the world model for the stoplight scenario.

Figure 2). The system will then mark state $\mathcal{A}$ as processed, pop state $\mathcal{B}$ off the stack, and derive the new successor state $\mathcal{C}$ via the temporal transition indicating that the light will change to yellow. Again an action is chosen for the current state, but this time the **cross-intersection** action is chosen because it is applicable (Green is safe to cross) and because it leads to the desired result. So at this point CIRCA has planned a simple reaction indicating that, when the light is green, the agent should cross. But the system has not yet shown why this action is guaranteed to be appropriate when executed; it has not yet addressed the sense/act gap, and the possibility that the light will change before the **cross-intersection** action is completed.

CIRCA addresses these issues by ensuring that the propositions used to satisfy the action's preconditions are covered by intervals of predictive sufficiency. The system knows the worst-case execution time of all of its sensing and action primitives, as well as their combinations. Thus the system knows exactly how long it will take, in the worst case, to detect the green light and cross the intersection (here, three seconds). To check for predictive sufficiency, the system must look for other domain processes that may be occurring during the action (i.e., transitions to other states). In this case, the system has recognized, based on domain knowledge, that there can be a temporal transition leading from the green state $\mathcal{B}$ to the yellow state $\mathcal{C}$ after a minimum of 25 seconds.

As noted above, CIRCA does not know how long the light has been green when it is observed; therefore, in the worst case, it is assumed that the temporal transition to the yellow state $\mathcal{C}$ occurs at the same time the system initiates the transition to cross the intersection. This corresponds to the "ghost" action transition in the figure (the dotted line), showing that the action planned for state $\mathcal{B}$ may actually be applied to state $\mathcal{C}$, leading to a new state $\mathcal{E}$ where the signal is yellow, but there is now a minimum of only two seconds before a temporal transition leads to a red light state.

In this process of looking at transitions out of the state for which the action is planned, CIRCA has shown that, although alternate results are possible, the precondition of the action ("safe2X") is known to hold for five seconds. This is the interval of predictive sufficiency: seeing a green light allows the system to guarantee at least five more seconds of safe crossing time. Because the process of sensing the green light and then crossing the street takes no more than three seconds, the interval of predictive sufficiency is long enough to cover the sense/act gap. Therefore, CIRCA can plan this action and guarantee that it will only be executed in appropriate situations*.

When CIRCA continues the planning process and tries to choose an action for the yellow state $\mathcal{C}$, it finds that the **cross-intersection** action is applicable and leads to the desired state. However, when the system tries to ensure that the "safe2X" precondition can be predicted to hold while the action is executed, it finds that a temporal transition leaving state $\mathcal{C}$ leads to the red state $\mathcal{A}$, which is "unsafe2X." Therefore, since the system does not know how much time may have passed in the yellow state $\mathcal{C}$ before the state was detected, and the subsequent state does not satisfy the action's preconditions, the action is rejected. In summary, CIRCA has used its explicit understanding of predictive sufficiency to derive a common rule of thumb used by drivers who glance at a traffic signal: if the light is green, go ahead and cross; if the light is yellow, do not start crossing, because the light may turn red too soon.

An interesting feature of this approach to avoiding inappropriate actions is that it requires no information about how frequently a particular sensory observation is being acquired— the example said nothing about how often the system checks to see if the light is green. If the system never even checks to see if the light is green, and thus never takes the **cross-intersection** action, it will never perform an inappropriate action. Clearly, this type of proof is only useful for goals that have no deadline. For real-time goals, that require response-time guarantees, this method is not sufficient.

To describe CIRCA's approach to meeting such real-time deadlines, we first introduce a more complex application domain.

The Puma Domain

The stoplight domain was used above for its intuitive simplicity; CIRCA has also been applied to a much larger robot control problem, illustrated by the simulation image in Figure 3. The Puma is assigned the task of packing parts arriving on the conveyor belt into the nearby box. Once at the end of the belt, each part remains motionless until the next part arrives, at which time it will be pushed off the end of the belt (unless the robot picks it up first). If a part falls off the belt because the robot does not pick it up in time, the system is considered to have failed. Thus, the arriving parts impose hard deadlines on the robot's responses; it must always pick up arriving parts before they fall off the conveyor.

The Puma is also responsible for reacting to an

*CIRCA currently only supports this test for preconditions that are required over the entire duration of an action.

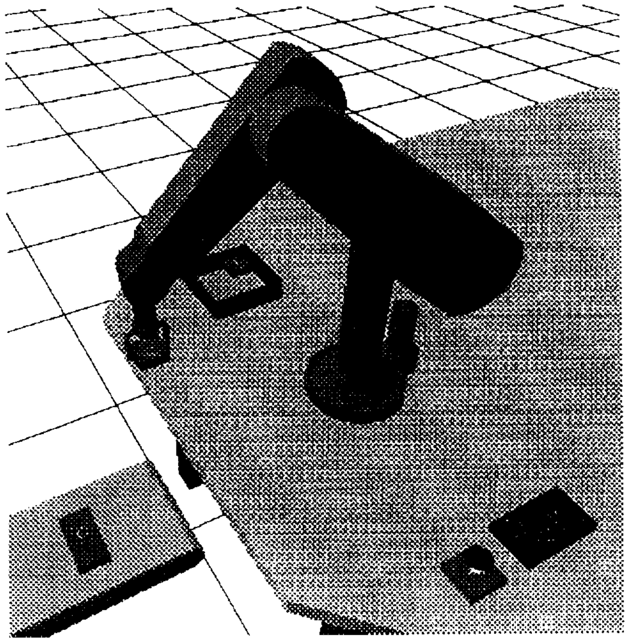


Figure 3: The Puma domain, with two hard real-time deadline constraints.

emergency alert light. If the light goes on, the system has only a limited time to push the button next to the light, or the system fails. This portion of the domain represents a completely asynchronous interrupt with a hard deadline on its service time.

Real-Time Response Guarantees

To deal with the hard deadlines in the Puma domain, the planning methods described above are not sufficient— they do not ensure that reactions will be timely, but rather that they will never be inappropriate. As we shall see, CIRCA must merge even more knowledge with its sensing information to guarantee timely responses that meet hard deadlines.

Figure 4 illustrates a small portion of the world model for the Puma domain[†], showing the representation of the hard deadline on picking up arriving parts. Parts are known to be spaced apart on the conveyor by at least some minimum distance. After a part arrives, the conveyor belt is considered to be "busy" for some amount of time (corresponding to the minimum part spacing) before the next part may arrive. Thus, from state $\mathcal{A}$ (where CONVEYOR-STATUS is BUSY) there is a temporal transition to state $\mathcal{B}$ (where CONVEYOR-STATUS is FREE), tagged with the value $min\Delta = 10$ (seconds) to indicate that state $\mathcal{A}$ must persist at least that long before the transition to state $\mathcal{B}$. From state $\mathcal{B}$, an event transition represents the fact that a part may arrive

[†]The full domain model includes more state features and hundreds of states and transitions.

at any time, leading to state $\mathcal{C}$. The potential failure resulting from the part falling off the conveyor is represented by the temporal transition out of state $\mathcal{C}$, also tagged with $min\Delta = 10$: if the next part arrives while this part is still on the conveyor, failure will occur.

To understand CIRCA's approach to making response-time guarantees, let us examine the planner's operation when it is considering state $\mathcal{C}$. The first phase of the planning process finds applicable event and temporal transitions, and recognizes that there is a potential temporal transition to failure. Since the failure is defined to be catastrophic, CIRCA realizes that it must preempt the temporal transition. That is, CIRCA decides it must execute some action that will definitely occur before the earliest time the temporal transition to failure can occur. A simple lookahead shows that the action **pickup-part-from-conveyor** will successfully avoid the failure. Now the only challenge is to ensure that the action will happen quickly enough. To ensure that the transition to failure is preempted, CIRCA commits to repeatedly executing a reaction that checks for the conditions of state $\mathcal{C}$ and implements the chosen action, at least frequently enough to ensure that the action will be completed before failure can occur. That is, CIRCA decides how quickly it must poll the sensors to detect the imminent failure and prevent it.

It is fairly obvious that, to guarantee that the system will simply detect the potential failure represented by state $\mathcal{C}$, which has a minimum possible duration ($mindur(P)$) of 10 seconds, CIRCA must test for the state at least once every 10 seconds. However, detecting the state $\mathcal{C}$ is not sufficient: the system must be able to ***finish*** the action of picking up the part before it can fall off the conveyor. In the terms introduced previously, the interval of predictive sufficiency during which the part is known to remain on the conveyor must cover the chosen action, in addition to its preconditions. To provide this predictive sufficiency, CIRCA relies on its additional knowledge about the frequency with which CIRCA itself will be obtaining sensory information. For example, if the period of the repeated observations is $\rho(O)$ seconds, then an observation in which the condition does hold, following an observation in which the condition does not hold, indicates that the change of state must have occurred in the last $\rho(O)$ seconds. Therefore, the condition must continue to hold for at least $mindur(P) - \rho(O)$ seconds.

Thus we have a modified interval of predictive sufficiency, based on both knowledge of the domain and knowledge about the ongoing performance of the reactive system itself. The AIS actually reasons about the performance of the reactive system it is designing to derive the predictive sufficiency of the observations it plans to make. To guarantee that every real-time reaction will be checked and executed before its corresponding deadline, CIRCA must show that the predictive sufficiency of the observations covers the sense/act gap and the duration of the chosen action. That is, $mindur(P) - \rho(O) > t_{\alpha\beta} - t_i$. In our Puma domain example, if the **pickup-part-from-conveyor** action takes 3 seconds, we have $10 - \rho(O) > 3$, so that $\rho(O) < 7$. If CIRCA can guarantee to execute the reaction that tests for state $\mathcal{C}$ and picks up the part at least once every 7 seconds, it can guarantee that it will not drop any parts off the conveyor[‡].

Making this reaction frequency guarantee is the job of CIRCA's Scheduler module (see Figure 1). The AIS uses the methods described above to derive frequency requirements for mission-critical reactions, and sends those reactions to the Scheduler. The Scheduler examines the capacity of the RTS to see if the available resources are sufficient to meet those requirements: if so, a schedule of reaction executions is returned to the AIS. If the RTS resources are not sufficient to guarantee the reaction rates specified by the AIS, the Scheduler will return an error message to the AIS, indicating that some performance tradeoff will be required in this overconstrained domain.

Knowledge Requirements

As we have noted, predictive sufficiency can only be established by combining immediate sensor information with additional knowledge about the domain. The basic form of the required knowledge is the "minimum duration" of some condition. That is, the system must know that some sensed state of the environment always persists for some minimum amount of time. In the stoplight domain, for example, the system must know the minimum duration of each signal color. In general, this type of knowledge might be acquired in one of two ways.

First, the system might have previous experience with the domain (or similar domains), and be able to extrapolate from that experience the requisite minimum durations. Experienced drivers know that no green light lasts for less than 5 seconds. Learning and past experience can thus play a key role in reasoning about predictive sufficiency.

Second, knowledge of minimum durations may also be derived from simple first principles, given precursor knowledge of the maximum rate of related (underlying) processes. For example, in the Puma domain, the minimum duration of the (CONVEYOR-STATUS

[‡]At least, not from this particular part of the state space.

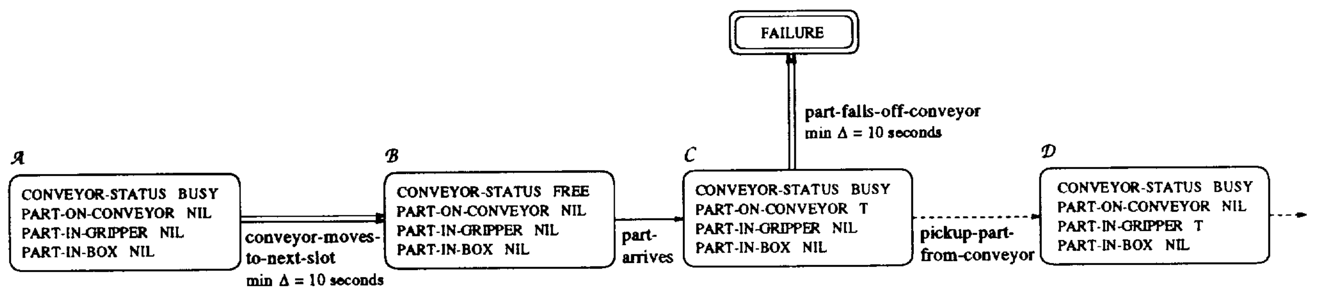


Figure 4: A small, abstracted portion of the Puma domain model.

BUSY) condition is determined by the maximum part arrival rate, which in turn is based on the conveyor belt speed and the spacing between parts. So if the system knows that parts must be at least ten inches apart and that the belt is moving at one inch per second, then the maximum part arrival rate is six parts per minute, and the minimum duration of the (CONVEYOR-STATUS BUSY) condition is ten seconds.

Currently, CIRCA makes no effort to learn minimum-duration knowledge itself, and it has only rudimentary, domain-specific methods to derive that knowledge from process rates. Instead, our focus has been on having CIRCA use that knowledge to reason about predictive sufficiency, and investigating the effects of explicitly dealing with the sense/act gap.

Conclusion

We have argued that all computing systems must make predictions about how the state of the world will evolve during the delay between sensing and action. The intuition behind the trend toward reactive systems has been that reducing this delay simplifies (but does not eliminate) prediction. In this paper, we have described how this intuition is really attempting to capture implicitly the concept of *predictive sufficiency*. By explicitly representing and reasoning about predictive sufficiency, we can determine exactly how long a gap between sensing and acting is allowable within a system, given its environment and its capabilities.

Predictive sufficiency is a critical concept for embedded agents, because it permits a system to make guarantees about its behaviors. We have shown how CIRCA implements predictive sufficiency to guarantee that it will not execute inappropriate actions and that it will react to its environment frequently enough to meet real-time deadlines.

Explicitly reasoning about predictive sufficiency also allows us to break away from the mind-set that decreasing the delay between sensing and acting is always desirable. Specifically, knowing the predictive sufficiency of an observation may allow a system to avoid some sensor polling by caching sensory data. No sensor readings need to be taken as long as a previous observation's interval of predictive sufficiency remains in force. We are investigating ways in which CIRCA can use its explicit knowledge of predictive sufficiency to design sensor caching schemes that maximize the use it gets out of each observation, reducing the frequency of costly observations without compromising the system's performance guarantees.

Our investigation of predictive sufficiency is a first step towards a more complete understanding of exactly when stored internal state is useful, and when it can lead to invalid predictions and failures. We hope to unify this approach with the epistemic proofs of Rosenschein and Kaelbling[11,12] to establish a full theory of the correspondence between a system's internal state, its predictions, and the world. This theory would allow strong prescriptive statements about when and how to use stored internal state.

References

[1] P. E. Agre and D. Chapman, "Pengi: An Implementation of a Theory of Activity," in *Proc. National Conf. on Artificial Intelligence*, pp. 268–272. Morgan Kaufmann, 1987.

[2] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–22, March 1986.

[3] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[4] R. J. Firby, "An Investigation into Reactive Planning in Complex Domains," in *Proc. National Conf. on Artificial Intelligence*, pp. 202–206, 1987.

[5] E. Gat, "On the Role of Stored Internal State in the Control of Autonomous Mobile Robots," *AI Magazine*, vol. 14, no. 1, pp. 64–73, Spring 1993.

[6] L. P. Kaelbling and S. J. Rosenschein, "Action and Planning in Embedded Agents," in *Robotics and Autonomous Systems 6*, pp. 35–48, 1990.

[7] T. J. Laffey, P. A. Cox, J. L. Schmidt, S. M. Kao, and J. Y. Read, "Real-Time Knowledge-Based Systems," *AI Magazine*, vol. 9, no. 1, pp. 27–45, 1988.

[8] D. J. Musliner, *CIRCA: The Cooperative Intelligent Real-Time Control Architecture*, PhD thesis, The University of Michigan, Ann Arbor, MI, September 1993. Also available as CSE-TR-175-93.

[9] D. J. Musliner, E. H. Durfee, and K. G. Shin, "CIRCA: A Cooperative Intelligent Real-Time Control Architecture," to appear in *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, no. 6, , 1993.

[10] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, CA., 1980.

[11] S. J. Rosenschein, "Synthesizing Information-Tracking Automata from Environment Descriptions," Technical Report 2, Teleos Research, July 1989.

[12] S. J. Rosenschein and L. P. Kaelbling, "The Synthesis of Digital Machines with Provable Epistemic Properties," in *Proc. Conf. Theoretical Aspects of Reasoning About Knowledge*, pp. 83–98, 1986.

[13] M. J. Schoppers, "Universal Plans for Reactive Robots in Unpredictable Environments," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 1039–1046, 1987.

[14] J. A. Stankovic, "Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems," *IEEE Computer*, vol. 21, no. 10, pp. 10–19, October 1988.

[15] D. Wilkins, "Domain-Independent Planning: Representation and Plan Generation," *Artificial Intelligence*, vol. 22, no. 3, pp. 269–301, April 1984.

53-63 [illegible] [illegible]

AIAA-94-1214-CP

USING GENERIC TOOL KITS TO BUILD INTELLIGENT SYSTEMS*

6060

N94- 30565

David J. Miller
Sandia National Laboratories
Albuquerque, New Mexico

Abstract

The Intelligent Systems and Robotics Center at Sandia National Laboratories is developing technologies for the automation of processes associated with environmental remediation and information-driven manufacturing. These technologies, which focus on automated planning and programming and sensor-based and model-based control, are used to build intelligent systems which are able to generate plans of action, program the necessary devices, and use sensors to react to changes in the environment. By automating tasks through the use of programmable devices tied to computer models which are augmented by sensing, requirements for faster, safer, and cheaper systems are being satisfied. However, because of the need for rapid cost-effective prototyping and multi-laboratory teaming, it is also necessary to define a consistent approach to the construction of controllers for such systems. As a result, the Generic Intelligent System Controller (GISC) concept has been developed.[1] This concept promotes the philosophy of producing generic tool kits which can be used and reused to build intelligent control systems.

Introduction

There have been many approaches taken in developing robotic control systems.[2,3,4,5,6,7,8,9,10,11,12,13,14,15] In examining these efforts, a common set of requirements can be derived. This set minimally includes such elements as fast servo-level response based on sensory inputs, trajectory planning based on world models of the tasks to be performed, and an extensible computing environment that supports asynchronous control with multi-tasking and multiprocessing. Therefore, any approach used for designing and implementing intelligent systems should support these requirements.

There also continues to be discussions within the user community about the need for guidelines or standards for robotic architectures. Because the primary purposes of standards are to save time and money when developing new systems and to facilitate integration of multi-supplier components, any standards adopted should reflect these goals. Also, because software is becoming the most critical component of complex intelligent systems, any potential standards should address the issues of how to make it easier and more cost-effective to develop software for new intelligent system applications.

The primary solution to this problem is software reusability. Although this may seem too simplistic, and many would argue that more encompassing standardization should be pursued, designing software for reuse is technically a very difficult task.[16] Also, because most software is developed within the context of a specific project, budgets and deadlines normally preclude developers from doing anything beyond the scope of the immediate task at hand. To overcome this dilemma, long-range thinking and planning need to be performed in order to encourage a philosophy of producing generic tool kits. Such tool kits, although developed within the context of a specific application, should transcend the application to provide reusable capabilities which reflect the common set of requirements for intelligent systems. Reuse makes subsequent applications easier to develop, thereby saving time and money. As a result, relatively complex systems with "standard" components can be developed as cost-effective solutions to difficult problems. Sandia is pursuing this philosophy in the development of the Generic Intelligent System Controller.

In this paper, we first describe the GISC approach to developing control systems. Then we discuss four different generic tool kits which have been developed in support of this approach. Next we illustrate how these tool kits can be integrated to build an intelligent robotic system, with particular emphasis on the development of a reusable generic subsystem to control any transport device such as a manipulator or CNC machine. Finally, we show how this system is utilized in two prototype applications.

An Approach to Building Intelligent Systems

The GISC concept was originally developed as part of the U.S. Department of Energy's Robotic Technology Development Program to design and implement prototype intelligent systems for performing hazardous operations. It is now being used for a variety of applications, including laboratory automation, painting of large structures, and agile machining.

GISC is communication oriented and is based on the premise that sophisticated intelligent system performance is achieved

*This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-94AL85000.

by coordinating a collection of semi-autonomous subsystems, each with complementary capabilities. Each subsystem has a well-defined command-and-control interface, and a supervisory control program coordinates the overall activities of the system through these subsystem interfaces. Individual subsystems may also possess real-time low-level control functions which can be performed autonomously and asynchronously. With the right combination of supervisor and subsystem capabilities, such an approach supports the implementation of model-based control and sensor integration within reusable software structures. This approach also promotes the use of modularity, distributed multi-processing environments, and standard commercial interfaces.

Generic Tool Kits

In order to build a GISC-based system, tools are needed for developing and integrating the supervisor and subsystems into a complete operational control system. Four such tool kits have been developed to provide a range of capabilities required at all levels of an intelligent system. These include:

1) the GENISAS tool kit which provides the communication facilities needed for the distributed supervisor/subsystem paradigm; [17]

2) the RIPE/RIPL tool kit which enables development of generic subsystems by providing object-oriented interfaces to intelligent system devices; [18]

3) the SMART tool kit which enables development of underlying control systems that provide the performance and flexibility for sensor-based control and teleoperation; [19]

4) the Sancho tool kit which provides for easily reconfigurable menu-based operator interfaces and a dynamic simulation environment.[20]

Figure 1. conceptually illustrates how a GISC-based system is organized with respect to these tool kits.

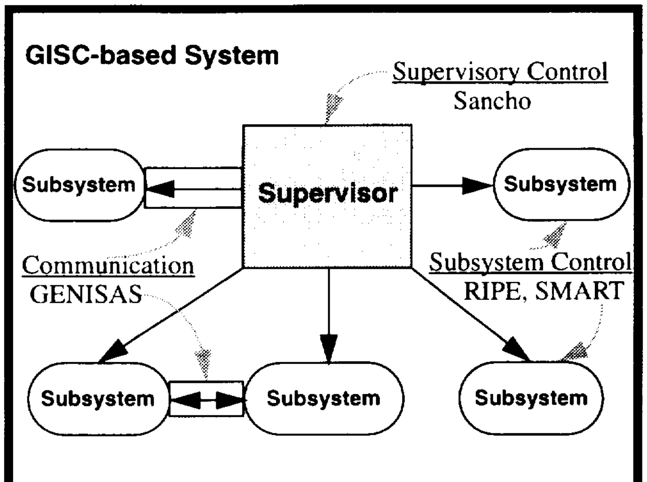


Figure 1. Reusable Tool Kits for Building GISC Systems

GENISAS - One of the key elements of any distributed intelligent system architecture is a powerful communication mechanism. The General Interface for Supervisor and Subsystems (GENISAS) is a client/server-based tool kit which provides general communication software interfaces between a supervisory control program and semi-autonomous subsystems, such as those which would be defined in a GISC-based system. There are four main components comprising the tool kit. The first component consists of low-level communication and utilities libraries which are provided to support reliable transmission of atomic messages and virtual multi-channels for commands, data, status, and exceptions. The next two components include supervisor (client-based) and subsystem (server-based) command and event processing libraries. Finally, there are facilities for message construction, parsing, and conversion. All of these libraries provide capabilities which allow the user to define command sets for table-driven command processing between supervisor and subsystem, data transfer requirements based on single point of control, events for asynchronous processing, and symbol manipulation.

The tool kit uses an object-oriented approach to define standard client and server base classes implemented in the C++ programming language. Through inheritance, application-specific subclasses can be derived. The base classes supply all of the supervisor-to-subsystem communication facilities. The subclasses, which are normally defined by the user, provide the specific command sets and command implementations for control of a particular subsystem, such as for a manipulator or sensor subsystem.

RIPE/RIPL - Another tool kit, the Robot Independent Programming Environment and Robot Independent Programming Language (RIPE/RIPL) , is the culmination of one of the earliest efforts to apply object-oriented technologies to building robotic software architectures. RIPE models the major components of a system as a set of C++ software classes. It consists of two main class inheritance hierarchies, *Device* and *CommunicationHandler*. The *Device* hierarchy contains subclasses for different kinds of devices normally found in an intelligent system. Active devices which have the property of being able to move or transport a tool or work piece are derived from the *Transport* subclass. Transport devices include robots, CNC machines, conveyors, translation tables, or autonomous vehicles. Passive devices, which are manipulated by the active devices, are derived from the *Tool* subclass, and *Tool* is further partitioned into particular types of tools such as *Sensor* or *Grabber*. The *CommunicationHandler* class hierarchy defines different ways of communicating with these devices, including serial, parallel, or network-based message passing. A clear separation is maintained between device class implementations and communication interfaces. Figure 2. illustrates the inheritance hierarchy for *Device*.

A generic set of object messages or "commands" are defined for each of the abstract base classes, and these messages constitute RIPL. For example, a generic set of RIPL calls is defined for the *Robot* class, and these commands are used for all robots. RIPL object messages are implemented as methods of the robot subclasses defined for each robot type. These subclass implementations serve as "translators" from the generic language to the robot-specific control environment. Implementations are obviously different for different vendors, but the interface is the same. Inheritance and polymorphism are used to associate these generic messages with each subclass defined for a particular robot type, thereby providing a mechanism for generically programming any robot for which a RIPE subclass has been implemented. The entire RIPE/RIPL tool kit is packaged as a set of class libraries.

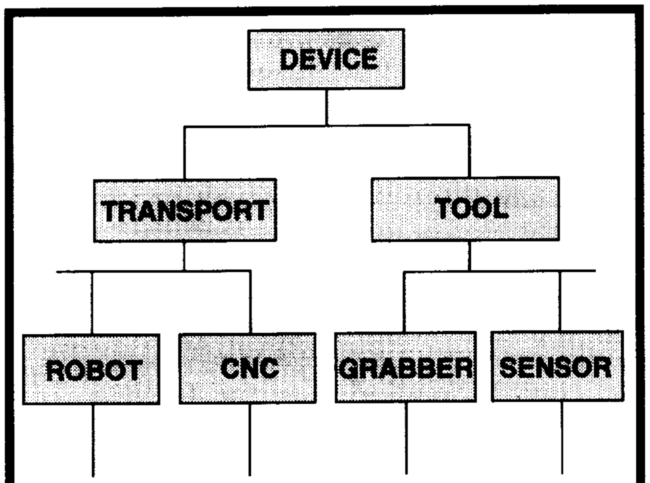


Figure 2. RIPE Class Inheritance Hierarchy

resent stiffness, and transformers represent Jacobians. Modules are connected to create a complete circuit which represents a control system. Typical modules include trajectory modules, kinematic modules, robot joint modules, sensor modules for force control and compliance, and input modules for space ball teleoperation or force reflection. Figure 3. illustrates a simple control system using three SMART modules. To use the tool kit, an application must define description files which indicate the number and types of modules to be used, how they are distributed, how information is passed between them, their period of operation, and appropriate filter constants.

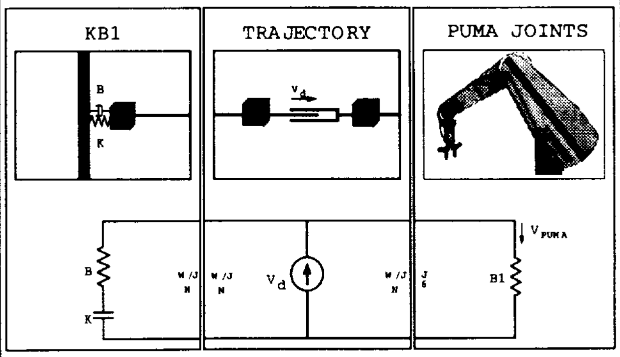


Figure 3. SMART Joint Controller for a PUMA 560

__SMART__ - For low-level control of actuators and sensors, a third tool kit called SMART (Sequential Modular Architecture for Robotics and Teleoperation) provides the capabilities required for stable autonomous and teleoperated closed loop feedback control. This tool kit can be used with any robot that is capable of accepting external position set points, and it can be used with any sensor that has a VME-based interface. The tool kit consists of a collection of C language libraries, each of which defines an interface to a distinct system "module" such as a sensor, actuator, input device, or kinematic/dynamic element. These "modules" can be asynchronously distributed across multiple CPUs and can execute in parallel with individual fixed-rate servo loops ranging from 100Hz to 1KHz.

SMART is based on 2-port network theory in which each module has a network equivalent. For example, inductors represent inertia, resistors represent damping, capacitors rep-

__Sancho__ - Sancho, a workstation-based tool kit, provides a GISC supervisory control program coupled with interface libraries which connect this supervisor to a graphical programming environment. This environment includes a menuing system based on X-Windows. Through these menus, an operator can command tasks and control the state of the system. Multiple active menu palettes allow for operations to be initiated in parallel. Communication objects from the GENISAS tool kit are used internally by the supervisory control program to connect it with an appropriate GISC subsystem such as a manipulator subsystem. Figure 4. shows an example of the graphical user interface for CNC machines as it appears to the operator on a Silicon Graphics workstation.

The functions performed by the menus are reconfigurable through ASCII file definitions, thereby allowing the supervisory control program to be reused for controlling different subsystems. A simulation interface library also provides facilities for the operator to execute a commercial simulation package such as Deneb's IGRIP. The operator can then interact with the work cell models that are loaded into this environment in conjunction with the menuing system and supervisor. The simulation environment is also linked through GENISAS to the real-time control system, providing for dynamic model updating and position tracking.

This requires the development of interfaces between the tool kits which allow them to maintain their autonomy and, at the same time, allow them to interact with each other according to the GISC philosophy. Such interfaces have been developed, and complete intelligent control systems have been implemented. These systems utilize the tool kits to perform tasks related to problems in such diverse areas as waste remediation and information-driven manufacturing.

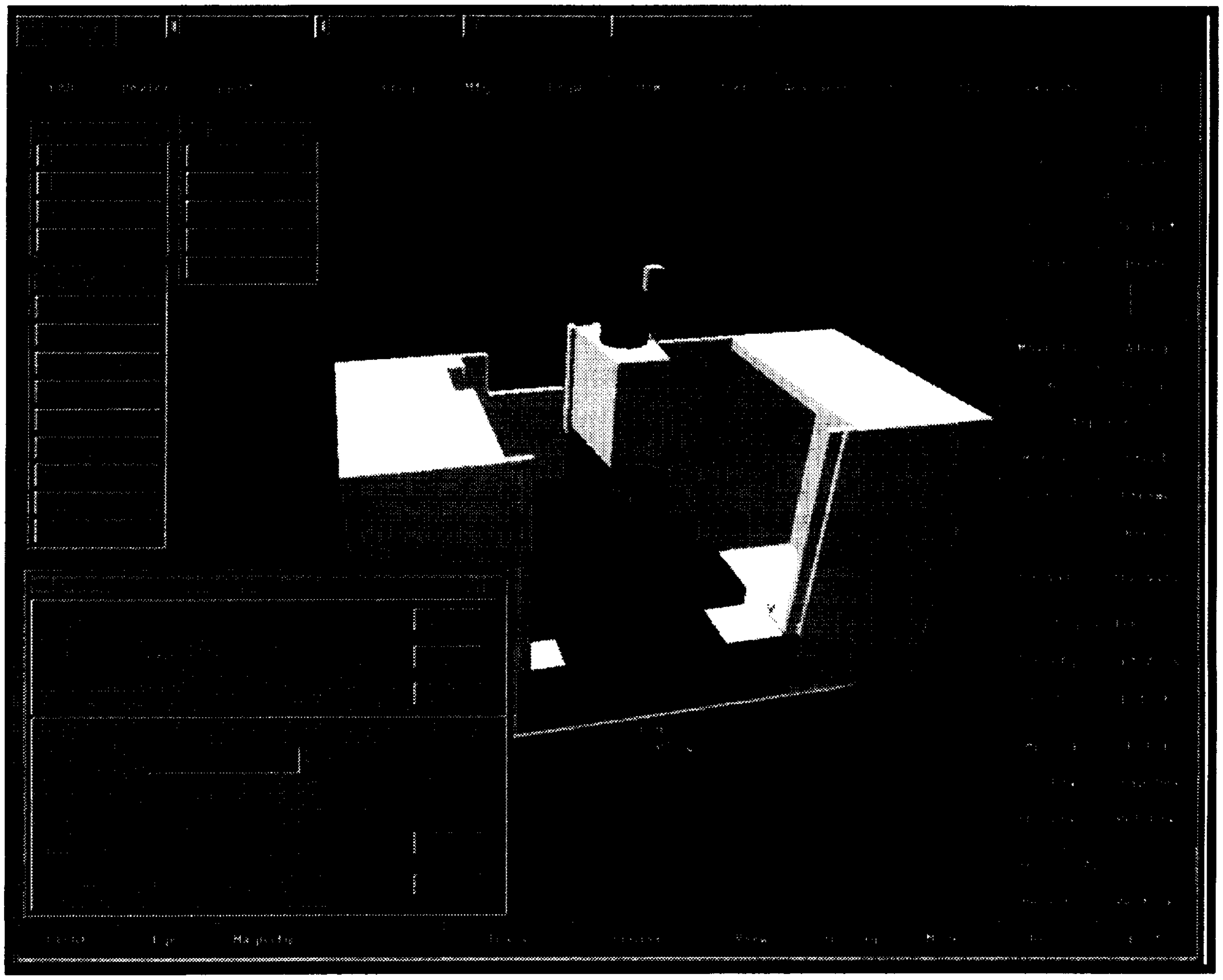


Figure 4. Sancho Graphical Programming Interface

Generic Tool Kit Interfaces
Each of these tool kits, aside from the supervisory control program, can be used completely independently of each other. This implies that they can be used and reused to implement robotic systems based on paradigms which are different from the GISC concept. On the other hand, by integrating them, a very powerful environment can be created for building intelligent system applications which are based on GISC.

Sancho to GENISAS Interface - Beginning at the operator interface level, the supervisory control program provided with Sancho automatically supplies an interface to the GENISAS tool kit because its function is to control the subsystems required for a particular application. This interface includes menu callback routines which use GENISAS client objects and their associated messages to communicate appropriate commands to the available subsystems. The set of commands, as reflected by the menuing system, may be application-specific. However, as mentioned previously, the command set can be easily changed through ASCII configuration data files. Similarly, the menuing system can be interactively redesigned in order to meet customer specifications. Both of these tailoring operations can be performed with minimal programming effort.

The other tools in Sancho provide an interface to Deneb's IGRIP simulation package which is simply treated as another GISC subsystem. If a different simulation package is selected for an application, then a new interface library must be implemented. However, the application programmer's interface between the supervisory control program, menuing system, and the simulation environment should remain the same. Only the underlying simulation interface library implementation must reflect the requirements of the particular simulation package used.

GENISAS to RIPE/RIPL Interface - The next required interface is between GENISAS and RIPE/RIPL. This interface occurs at the subsystem level and is relatively straightforward since both tool kits are object-oriented and implemented as C++ class libraries. A GISC subsystem is normally controlled by a server process which is defined as a subclass of the GENISAS *StdServerProcess* base class. It therefore inherits all of the communication facilities required by any server. This subclass also defines the methods which implement the command set associated with the subsystem it services. These methods, in turn, are implemented by using RIPL methods defined for the device or devices controlled by the subsystem. The integrated use of RIPE with GENISAS allows for distribution of RIPE objects across multiple CPUs and environments, and provides an ASCII-based script file interface which translates into C++-based RIPL methods.

RIPE/RIPL to SMART Interface - The interface between RIPE/RIPL and SMART is somewhat complex due to the asynchronous, distributed nature of the underlying SMART modules. This interface has two primary components, one associated with the server subclass and one associated with the RIPL methods used by the server. Normally when a subsystem is booted which uses SMART, the desired SMART modules are automatically downloaded as part of a startup script, and numerous tasks associated with them are spawned. The number, type, and distribution of modules are determined by configuration files which are currently compiled with the subsystem initialization code. If multiple CPUs are utilized by SMART, an exact copy of the server code is downloaded to each CPU. These servers are started after SMART module initialization is completed. They also use configuration files to build a "roadmap" which indicates where the SMART modules are located. Through data-driven logic, the server on the first CPU behaves as a "traffic cop" by directing commands received from the supervisor to either itself or to the other servers according to where the SMART modules are located and according to which modules are required to carry out each command. Note that the server code does not have to be modified for different SMART configurations. Only the ASCII configuration files need to be changed. This essentially comprises the first interface to SMART.

The second interface is simpler. The RIPL methods used by the server to carry out commands call routines from the SMART tool kit. These routines, in turn, cause the asynchronous control tasks to change state and thereby affect the state of the devices being controlled by the subsystem. However, a problem with this approach is that RIPL methods now appear to be directly tied to the SMART tool kit rather than remaining autonomous. This can be prevented by defining a *SMARTRobot* class in RIPE which isolates the RIPL methods that must be implemented in terms of the SMART tool kit. Then subclasses can be derived from *SMARTRobot* for particular robot types. These subclasses can inherit either a standard robot interface or the SMART robot interface. Therefore, only the *SMARTRobot* class is dependent upon the SMART tool kit.

Generic Subsystem for Transport Devices

Using the interface templates just described, a generic server subsystem has been implemented which can be reused with minor modifications to control any transport device that has a RIPL translator. A generic command set has been defined for this transport subsystem, thereby eliminating the need to reconfigure the Sancho interface whenever a different manipulator is required for a new intelligent system application. Brief descriptions of the generic commands are given in Figure 5.

During a graphical programming session using Sancho, these commands are sent to the generic server subsystem by a GENISAS client which is contained within the supervisory control program. They are sent as ASCII strings with variable numbers of arguments and argument formats. GENISAS internally handles the parsing of the commands and their arguments to determine which method in the server subsystem should be invoked to carry out the command.

The generic transport subsystem is defined as a *RobotServer* subclass of the GENISAS *StdServerProcess* base class. It therefore inherits all of the communication facilities required by any server. The *RobotServer* subclass itself contains the methods which implement the generic command set. These methods, in turn, are implemented by using RIPL methods defined for the appropriate RIPE device driver subclass. This is accomplished by defining a generic pointer (*ptr_robot*) to the RIPE subclass inside *RobotServer* and establishing a containment relationship between them. Whenever a *RobotServer* object is created during subsystem initialization, the *RobotServer* constructor will create the appropriate RIPE object or objects for the transport device in use. This, in turn, provides the initialization for the device so that it is ready to be controlled through the generic commands.

Lock:	give supervisor exclusive REMOTE control
Release:	give subsystem exclusive LOCAL control
Activate:	place transport device in an active state
Deactivate:	place transport device in an inactive state
Configure:	configure subsystem for subsequent cmds
SetUnits:	set the linear and/or angular units
SetSpeed:	set the absolute speed
SetAcceleration:	set the absolute acceleration
SetToolLength:	set the tool length for the current tool
ReportState:	return the current device state
MoveTo:	perform a motion in world space
MovebyJoint:	perform a motion in joint space
MoveReact:	move until a sensor threshhold is exceeded
MoveComply:	move while complying to a surface
ManualControl:	move under control of a teleoperated device
LoadPath:	download a path segment to a motion queue
MoveAlongPath:	perform a path move using current queue
ClearPath:	clear path motion queue
StopMotion:	stop current motion gracefully
GetTool:	get specified tool
PutTool:	put specified tool
OpenGripper:	enact motion for current tool (open jaws)
CloseGripper:	enact motion for current tool (close jaws)
InitRecordFile:	record a log of subsequent trajectories
CloseRecordFile:	stop recording trajectories

Figure 5. Generic Transport Subsystem Commands

The *RobotServer* generic command implementations are identical for any transport device because all RIPE transport device subclasses use the same RIPL calls to program their associated hardware. An example of a simple template for the *RobotServer* method which implements the **Activate** command is shown in Figure 6. In this code, the server first determines which CPU the command should be executed on if the control system is distributed across multiple CPUs. If this particular copy of the server resides on CPU 0, which is by convention the CPU that the supervisor communicates with, then message routing must be handled correctly. *RobotServer* on CPU 0 uses an internal GENISAS client to ship the command to another copy of *RobotServer* on a different CPU if the command must be executed somewhere other than CPU 0.

The command is actually executed by calling RIPL method *change_state*. This method will somehow interact with the device to place it in an active state. For a SMART-based controller, this involves calling SMART library routines for activating the SMART control system. As long as each RIPE subclass required by the server has the standard RIPL calls, such as *change_state* for activating the transport device, the same implementation can be used by any server for any transport device. Note in Figure 6. how the *change_state* method is called using the generic *ptr_robot*. Therefore, for each different transport server implementation, the only code modifications required are redefinition of this pointer for the desired RIPE device object contained in *RobotServer* and substitution of the correct RIPE constructor call used to initialize that device. In other words, for a subsystem that controls a Puma robot, *RobotServer* will define a containment relationship with the RIPE class *PRobot*, and the generic *ptr_robot* will be initialized to point to a *PRobot* object. Likewise, for a subsystem that controls a CNC machine, *RobotServer* will define a containment relationship with RIPE class *CNCMachine*, and the generic *ptr_robot* will be initialized to point to a *CNCMachine* object. All of the *RobotServer* command methods will remain unchanged from subsystem to subsystem, producing a high degree of software reuse.

Application-specific information is maintained in ASCII configuration files which are accessed by the *RobotServer* constructor. Such information includes network configuration information, tool and sensor tables, and SMART configuration information if the SMART tool kit is being used for low-level control. The SMART configuration includes which SMART modules are required, which CPUs they are resident on, and which modules are accessed for each generic command implementation.

```
int RobotServer::Activate(int argc, void ** argv, char *e_msg) {
        int ret = OK ;
        static char fname[] = "Activate";
        int location ;
        char cntlCmdMsgCopy[100] ;

        entering(fname);

        // Determine where the command should be executed
        location = WhichCPU(fname) ;

        // If this is the main server and the command is to be executed
        // somewhere else, send the command to the appropriate cpu.
        // If the transmission is successful, also execute the command
        // on the main server to update state variables
        if ((location > my_cpu_number) && (my_cpu_number == 0))
        {
           sprintf(cntlCmdMsgCopy, "%s", fname) ;
           ret = clientP[location]->SendCommand(cntlCmdMsgCopy, e_msg) ;
           if (ret == OK)
              ret = ptr_robot->change_state(ACTIVATE) ;
        }

        // If this is the correct cpu, execute the command
        else if (location == my_cpu_number)
           ret = ptr_robot->change_state(ACTIVATE) ;

        // This server is not suppposed to execute the command
        else
           ret = ERROR ;

        return(ret);
}
```

Figure 6. Sample Code for a Generic Command Method

Currently this generic server is used to control several different manipulators and a CNC milling machine. Extension of the generic tool kits to support other devices is a straightforward, methodical process because existing detailed designs can be reused. For example, to support a new manipulator, a

RIPE subclass must be implemented which provides the translation from RIPL commands to corresponding hardware signals that produce motion. Because the RIPL interface design is already well-defined, the process basically involves implementing each of the methods associated with the RIPL command interface. Then a new version of the generic transport subsystem can be cloned which utilizes this new RIPE object to control the new manipulator. A similar scenario can be followed for extending the SMART tool kit. Development effort may still be significant since different devices have different interfaces with varying degrees of complexity. However, the amount of reuse and resultant savings in time and cost are also significant.

Applications

Complete intelligent control systems have been implemented which utilize all four tool kits and their interfaces to perform several prototype applications for environmental remediation and information-driven manufacturing. The resulting systems are based on the interactive menuing interface and simulation environment from the Sancho tool kit for automated planning and programming. The supervisory control programs use the set of generic commands described previously to control a transport device required by a given subsystem. This command set is easily extended or modified through Sancho ASCII configuration files and new *RobotServer* methods to reflect changing requirements. The generic transport server subsystem defined by subclass *RobotServer* is used to control either a manipulator or CNC machine. This subsystem connects to the supervisor through GENISAS and executes the generic commands for any manipulator or CNC machine that is supported by the RIPE/RIPL and/or SMART tool kits. Currently this includes a Schilling Titan2 manipulator, a Schilling ESM long reach manipulator, various models of the Puma robot, and a Fadal vertical machining center. By starting out with this base system, task-level programming can be accomplished by generating scripts containing sequences of generic commands that perform useful operations.

Underground Storage Tank Remediation

One application for environmental remediation involves the clean up of waste sites in which human exposure to radiation or other hazardous elements is unacceptable. Traditional manual master-slave methods for performing such remote operations have very low productivity and consequently a very high cost. Therefore, systems which use automated planning and programming and sensor-based and model-based control to perform these operations are faster, safer, and cheaper.

One of the tasks which has been implemented using this system is the cutting and removal of structures such as pipes from underground storage tanks. A Schilling Titan2 manipulator is used to perform the task. The operator first commands the manipulator to pick up a hydraulic cutter end effector and approach a pipe under graphical control, based on a model of the tank environment. The operator uses a mouse to select any point along the pipe where he wishes to perform the cut. Using knowledge of the location and orientation of the pipe in the graphical model as well as knowledge of approved pipe shearing practices, the control system automatically computes the correct motions to position the cutter approximately one foot from the pipe surface. This approach can be simulated first and previewed by the operator to verify that it can be executed safely. Once the manipulator is near the pipe, the operator can then command the system to perform a docking operation using ultrasonic sensors to center the pipe within the jaws of the cutter. Once docked, the operator commands the cutter to shear off the pipe, followed by an undocking operation.

All of the manipulator motions are executed through the generic robot server using the generic command set. Additional subsystems are used in GISC-like fashion to control the sensors and the cutter. The docking operation is therefore actually a "macro" command which consists of a sequence of generic commands to perform compliant motion. This macro is an example of how application-specific software can be developed within the context of the generic control system to perform specific tasks.

Intelligent CNC Architecture for Agile Machining

Another application in the area of information-driven manufacturing involves the development of an intelligent CNC machine control system architecture which enables one to more fully automate the process from CAD design to finished part. The software implementation once again consists of the graphical programming environment coupled with the generic transport subsystem which controls a Fadal Inc. vertical machining center through a RIPL translator. The Fadal machine encoders are interfaced to the subsystem for real-time position tracking. In addition, a touch probe and structured lighting system are also interfaced to the subsystem for part and fixture location.

A typical scenario for using the system would begin with the operator opening a window onto his favorite CAD system and designing a part containing features which require machining. When the design is completed, CAD models for the finished part, raw stock, and fixtures are imported into a simulation environment such as Deneb's IGRIP. A kinematically correct model of the milling machine is available within this environment, and the operator performs the necessary setup of the virtual machine by interactively arranging the CAD models of the parts and fixtures in an optimal way for machining operations. The operator then interactively generates a tool path by using a space ball to maneuver the machine tool around the part. The system automatically records the motions which can be played back in a simulation mode to verify that there are no collisions and that an acceptable material removal sequence is being performed. When the operator has completed the generation of the program, he can then mount the actual parts and fixtures onto the selected machine bed and use a sensor such as the touch probe to locate the parts and fixtures with respect to the machine coordinate system. This information can be uploaded to the graphical programming environment which uses it to perform its own calibration process to accurately register the model with the real physical world. Then the tool paths derived from the previous simulation are automatically adjusted based on this calibration. Finally, the graphically generated program is downloaded to the generic transport subsystem and executed as a sequence of generic commands to machine the part.

Summary

In summary, rapid, cost-effective deployment of intelligent systems to perform useful operations requires a software infrastructure which allows a system builder to immediately focus on the application-specific requirements of the task. Such an infrastructure is best provided through a set of complementary, integrated generic tool kits which serve as the building blocks for new application development. Such tool kits should provide the necessary communication, device, and operator interfaces within reusable software structures. As standalone products, they are independent of any particular application, but in the hands of the system integrator, they can be used to build very powerful intelligent systems for a variety of automated tasks.

As generic tool kits proliferate and are made more robust and easier to utilize, then de facto standards may evolve for intelligent systems which are based on common interfaces established within these tool kits. Obviously, there are many barriers to overcome in terms of defining these interfaces and learning how to develop truly reusable code. Technology transfer and commercialization of these packages is also essential in order to establish a market-driven standardization climate. Companies such as Adept, Schilling, PAR Systems, and Trellis are already developing and marketing more open, modular approaches to control systems due to repeated requests from the robotics R&D community. With continued efforts within this community to define the necessary interfaces and then transfer them to the commercial sector, we may gradually see an evolution toward the availability of standard tool kits which can be used to construct whatever kind of intelligent robotic system is needed for future applications.

References

1. Griesmeyer, J. M., McDonald, M. J., Harrigan, R. W., Butler, P. L. , and Rigdon, B., "Generic Intelligent System Controller (GISC)," *Sandia Internal Report, SAND92-2159*, Sandia National Laboratories-New Mexico, October, 1992.

2. Albus, J. S., McCain, H. G., Lumia, R., "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," *NIST Technical Note 1235*, April 1989.

3. Backes, P., Hayati, S., Hayward, V., and Tso, K., "The KALI Multi-arm Robot Programming and Control Environment," *Proc. of NASA Conf. on Space Telerobotics*, January 31-February 2, 1989.

4. Brooks, Rodney A., "Elephants Don't Play Chess," *Robotics and Autonomous Systems*, No. 6, pp. 3-15, 1990.

5. Dilts, D. M., Boyd, N. P., Whorms, H. H., "The Evolution of Control Architectures for Automated Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 10, No. 1, pp. 79-93, 1991.

6. Elfving, A., Kirchhoff, U., "Design Methodology for Space Automation and Robotics Systems," *ESA Journal*, Vol. 15, 1991.

7. Hayati, S., Venkataraman, S. T., "Design and Implementation of a Robot Control System with Traded and Shared Control Capability," *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1310-15, Scottsdale, AZ, May 14-19, 1989.

8. Hayes-Roth, F., Erman, L. D., Terry, A., Hayes-Roth, B., "Domain-Specific Software Architectures: Distributed Intelligent Control and Communication," *IEEE Symposium on Computer-Aided Control System Design*, pp. 117-128, Napa, CA, March 1992.

9. Hormann, A., "A Petri Net Based Control Architecture for a Multi-Robot System," *Proceedings. IEEE International Symposium on Intelligent Control 1989*, pp. 493-8, Albany, NY, Sept. 25-26, 1989.

10. Martin Marietta, "Draft Volume I of Next Generation Workstation/Machine Controller (NGC) Specification for an Open System Architecture Standard (SOSAS)," *Document No. NGC-0001-13-000-SYS*, March 1992.

11. Mitchell, T.M., "Becoming Increasingly Reactive," *AAAI-90 Proceedings: Eighth National Conference on Artificial Intelligence*, Vol. 2, pp. 1051-8, Boston, MA, July 29-Aug. 3, 1990.

12. Rossol, Lothar, "Nomad Open Architecture Motion Control Software," *Proceedings of the International Robots & Vision Automation Conference*, Detroit, Michigan, April 5-8, 1993.

13. Saridis, G. N., "Architectures for Intelligent Controls," *Symposium on Implicit and Nonlinear Systems*, Ft. Worth, TX, December 14-15, 1992.

14. Sorensen, Steve, "Overview of a Modular, Industry Stan-

dards Based, Open Architecture Machine Controller," *Proceedings of the International Robots & Vision Automation Conference*, Detroit, Michigan, April 5-8, 1993.

15. Stewart, D. B., Volpe, R. A., Khosla, P. K., "Integration of Real-Time Software Modules for Reconfigurable Sensor-Based Control Systems," *Proceedings 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 7-10, 1992.

16. Freeman, P., "Tutorial: Software Reusability," *IEEE Computer Society Press*, ISBN 0-8186-0750-5, 1989.

17. Griesmeyer, J. M., "General Interface for Supervisor and Subsystems (GENISAS)," *Sandia Internal Report*, Sandia National Laboratories-New Mexico, October, 1992..

18. Miller, D. J. and Lennox, R. C., "An Object-Oriented Environment for Robot System Architectures," *IEEE Control Systems*, Vol. 11, No. 2, February 1991.

19. Anderson, R. J., "SMART: A Modular Architecture for Robotics and Teleoperation," *International Symposium on Robotics and Manufacturing (ISRAM '93)*, Santa Fe, NM, April, 1993.

20. McDonald, M. J. and Palmquist, R. D., "Graphical Programming: On-Line Robot Simulation for Telerobotic Control," *Proceedings of the International Robots & Vision Automation Conference*, Detroit, Michigan, April 5-8, 1993.

606 AIAA-94-1218-CP

A REACTIVE SYSTEM FOR OPEN TERRAIN NAVIGATION: PERFORMANCE AND LIMITATIONS

N94- 30566

D. Langer, J. Rosenblatt, M. Hebert
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213

[illegible]

P. 7

Abstract

We describe a core system for autonomous navigation in outdoor natural terrain. The system consists of three parts: a perception module which processes range images to identify untraversable regions of the terrain, a local map management module which maintains a representation of the environment in the vicinity of the vehicle, and a planning module which issues commands to the vehicle controller. Our approach is to use the concept of "early traversability evaluation," and on the use of reactive planning for generating commands to drive the vehicle. We argue that our approach leads to a robust and efficient navigation system. We illustrate our approach by an experiment in which a vehicle travelled autonomously for one kilometer through unmapped cross-country terrain.

1 Introduction

Autonomous navigation missions through unmapped open terrain are critical in many applications of outdoor mobile robots. To successfully complete such missions, a mobile robot system needs to be equipped with reliable perception and navigation systems capable of sensing the environment, of building environment models, and of planning safe paths through the terrain. In that respect, autonomous cross-country navigation imposes two special challenges in the design of the perception system. First, the perception must be able to deal with very rugged terrain. Second, the perception system must be able to reliably process a large number of data sets over a long period of time.

Several approaches have been proposed to address these problems. Autonomous traverse of rugged outdoor terrain has been demonstrated as part of the ALV [11] and UGV [10] projects. JPL's Robby used stereo vision [9] as the basis of its perception system and has been demonstrated over a 100 m traverse in outdoor terrain. Other efforts include: France's VAP project which is also based on stereo vision [2]; the MIT rovers which rely on simple sensing modalities [1]. Most of these perception systems use range images, from active ranging sensors or passive stereo, and build a map of the terrain around or in front of the vehicle. The planning systems use the maps to generate trajectories. The approaches used in the existing planning systems range from purely reactive to fully proactive, depending on the type of maps. The main questions in building such systems are: What should be in the map, and when should the map be computed?

In this paper, we argue that relatively simple methods of obstacle detection and local map building are sufficient for cross-country navigation. Furthermore, when used as input to a reactive planner, the vehicle is capable of safely traveling at significantly faster speeds than would be possible with a system that planned an optimal path through a detailed, high-resolution terrain map. Moreover, we argue that an accurate map is not necessary because the vehicle can safely traverse relatively large variations of terrain surface. For these reasons, we propose an approach based on "early evaluation of traversability" in which the output of the perception system is a set of untraversable terrain regions used by a planning module to drive the vehicle. The system relies on "early evaluation" because the perception module classifies regions of the terrain as traversable or untraversable as soon as a new image is taken. As we will show, early traversability evaluation allows for a more reactive approach to planning in which steering directions and speed updates are generated rapidly and in which the vehicle can respond to dangerous situations in a more robust and more timely manner.

The goal of this paper is to present and discuss the performance of the overall system. We start by giving an overview of the approach and of the system architecture in Section 2; we then describe the performance of the system in an actual experiment in Section 3. We focus on the individual components of the system in Sections 4 to 6. More detailed descriptions of the components may be found in [5] for the local map module, [12] for the planning component, and in [8] for the complete system description.

2 Early Evaluation of Traversability: Overview

The perception and navigation system was developed as part of the Unmanned Ground Vehicle (UGV) project. The support vehicle is a retrofitted HMWVV suitable for cross-country navigation (Figure 1). The sensor is the Erim laser range finder which acquires 64x256 range images at 2 Hz. An estimate of vehicle position is available at all times by combining readings from an INS system and from encoders. The goal of this system is to enable the vehicle to travel through unmapped rugged terrain at moderate speeds, typically two to three meters per second.

Because of the speed requirement, the perception system must update the local terrain map fast enough to keep up with vehicle motion. For that reason, it is impractical to build a detailed, high-resolution terrain map every time a new image is taken. Moreover, an accurate map is not necessary because the vehicle can safely tolerate relatively large variation of terrain surface. For these reasons, we used in this example an approach based on "early evaluation of traversability" in which the output of the perception system is a set of untraversable terrain regions which is used by a planning module to drive the vehicle. Untraversable regions are terrain features such as high slopes, ditches, or tall objects which would endanger the vehicle. The system relies on "early evaluation" because the perception module classifies regions of the terrain as traversable or untraversable as soon as a new image is taken. This has the advantage of reducing the amount of data passed to the planner for path generation and reducing the amount of computation needed in later stages of planning.

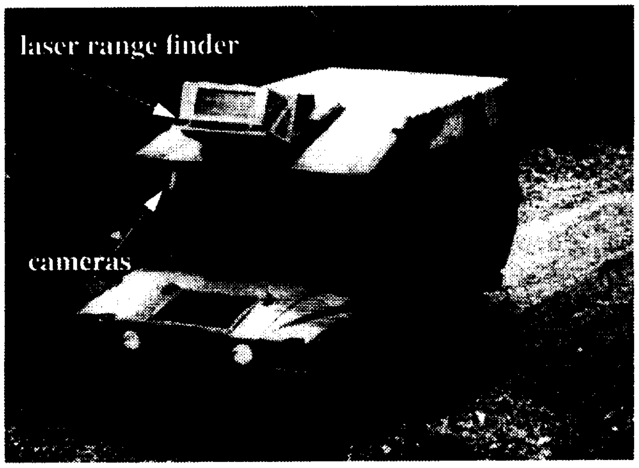


Figure 1: The testbed vehicle.

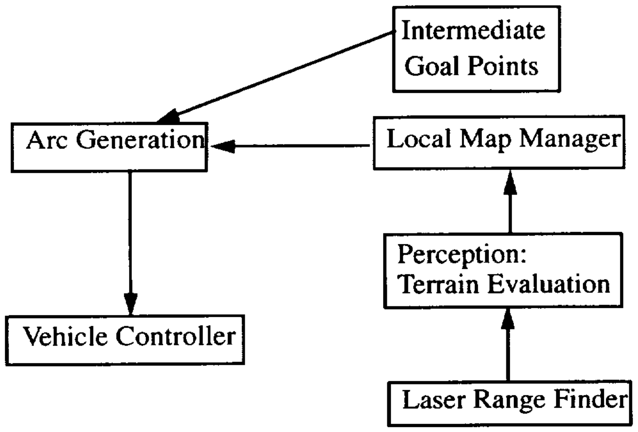


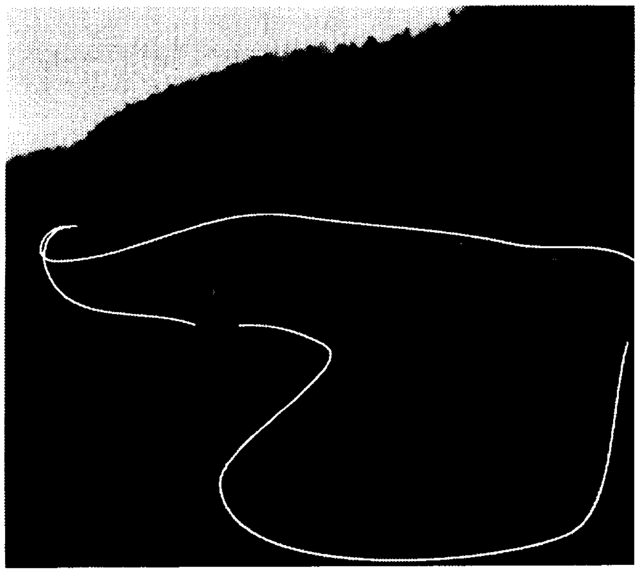
Figure 2: Architecture of the navigation system.

Figure 2 summarizes the system developed based on the idea of "early traversability". The perception component of the system consists of a terrain evaluation module which takes images from a range scanner and outputs untraversable regions to a local map manager. The local map manager maintains a consistent description of the terrain around the vehicle as it travels and send periodically a description of the untraversable regions in the vicinity of the vehicle to an arc generation module. The arc generation module rates each arc out of a finite set of arcs between forbidden if the arc hits an obstacle and clear if the arc does not pass close to any obstacle region. The arc generation module generates traversability votes for each of the arcs rather than the best arc to follow next. This permits the combination of these votes with votes from other modules. For example, we have used a goal-seeking module which steers the vehicle toward the next goal point. In practice, any navigation module could be substituted to the goal-seeking module.

3 System Operation: A Typical Mission

Figure 3 and Figure 4 show a typical run of the perception and navigation system. Figure 3 (a) shows the environment used in this experiment. The terrain includes hills, rocks, and ditches. The white line superimposed on the image of the terrain shows the approximate path of the vehicle through this environment. The path was drawn manually for illustrative purpose. Figure 3 (b) shows the actual path recorded during the experiment projected on the average ground plane. In addition to the path, Figure 3 (b) shows the obstacle regions as black dots and the intermediate goal points as small circles. In this example, the vehicle completed a one kilometer loop without manual intervention at an average speed of 2 m/s. The input to the system was a set of 10 waypoints separated by about one hundred meters on average. Except for the waypoints, the system does not have any previous knowledge of the terrain. Local navigation is performed by computing steering directions based on the locations of untraversable regions in the terrain found in the range images. An estimated 800 images were processed during this particular run.

Figure 4 shows close-ups of three sections of the loop of Figure 3. The black lines show the approximate paths followed by the vehicle in these three sections. Figure 5 shows the elevation map obtained by pasting together the images taken along the paths. In each figure, the grey polygons are the projections of the fields of view on the ground, the curved grey line is the path of the vehicle on the ground, and the white dots indicate locations at which images were taken. The images are separated by approximately two meters in this case. The paths shown in Figure 5 are the actual paths followed by the vehicle. It is important to note that these maps are included for display purposes only and that the combined elevation maps are not actually used in the system. Finally, Figure 6 shows displays of the local map which is maintained at all times around the vehicle. The squares correspond to 40x40 cm patches of terrain classified as untraversable regions or obstacles. These local maps are computed from the positions shown in Figure 4 and Figure 5 by the white arrows. The trajectories are planned using this compact representation rather than the detailed maps of Figure 5.


(a) View of terrain and approximate path.

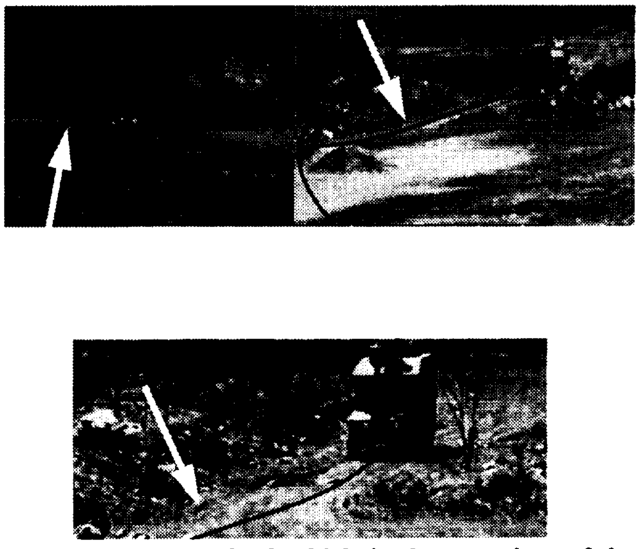
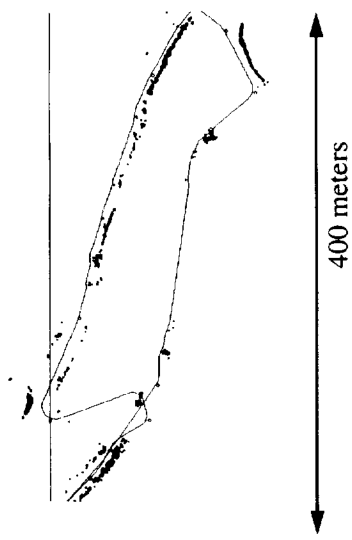


Figure 4: Local path of vehicle in three sections of the loop of Figure 3. The arrows indicate the locations at which the local maps are displayed in Figure 5 below.



(b) Exact path of vehicle; the obstacle regions are shown as black dots; the interme-

Figure 3: A Loop through natural terrain.

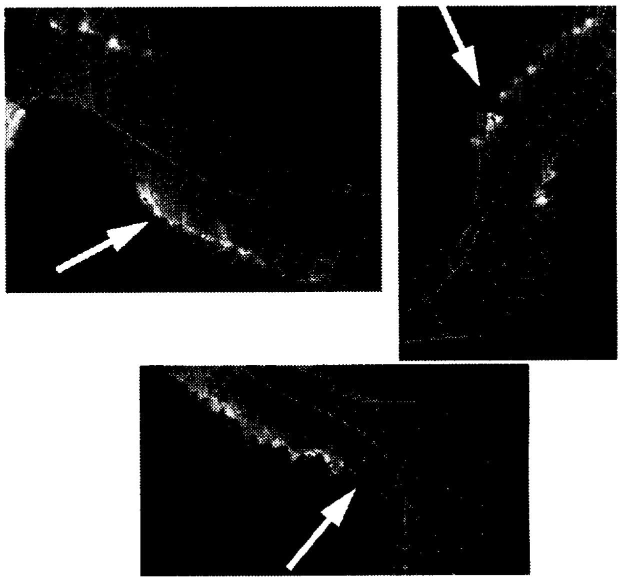


Figure 5: Display of the terrain as elevation maps for the sections shown in Figure 4. The polygons indicate the projection of the field of view of the sensor on the ground. The white line shows the path followed by the vehicle in this section. The white dots show the positions at which the images were taken. The arrows are placed at the same locations as in Figure 4.

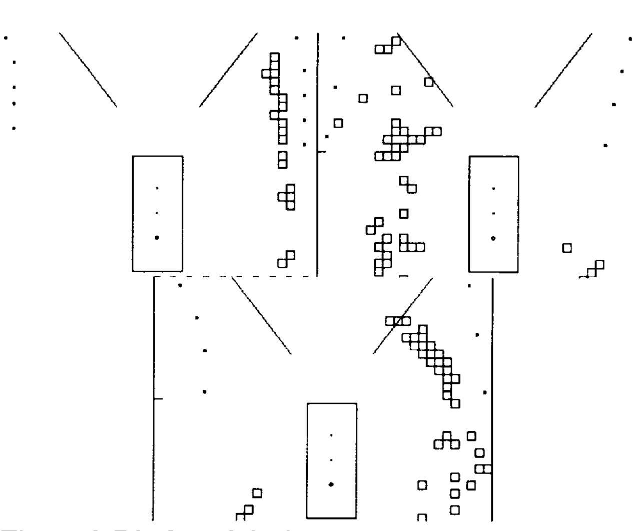
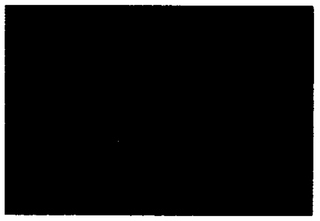


Figure 6: Display of the local traversability map at the locations marked by arrows in Figure 4 and Figure 5. Only the portion of the map in the immediate vicinity of the vehicle is displayed here. The vehicle is depicted by a rectangle. The untraversable regions are shown as squares.

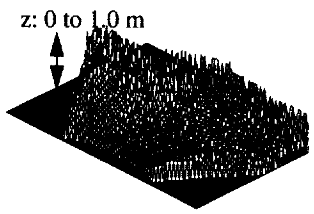
4 Perception

The range image processing module takes a single image as input and outputs a list of regions which are untraversable. After filtering the input image, the module computes the (x,y,z) location of every pixel in the range image in a coordinate system relative to the vehicle's current position. The coordinate system is defined so that the z axis is vertical with respect to the ground plane, and the y axis is pointing in the direction of travel of the vehicle. It is convenient to center the coordinate at the point used as the origin for vehicle control, in this case between the two rear wheels, rather than at the origin of the sensor. The transformation takes into account the orientation of the vehicle read from an INS system. The points are then mapped into a discrete grid on the (x,y) plane. Each cell of the grid contains the list of the (x,y,z) coordinates of the points which fall within the bounds of the cell in x and y. The size of a cell in the current system is 20 cm in both x and y. This number depends on the angular resolution of the sensor, in this case 0.5°, and on the size of terrain features which need to be detected. The terrain classification as traversable or untraversable is first performed in every cell individually. The criteria used for the classification are the height variation of the terrain within the cell, the orientation of the vector normal to the path of terrain contained in the cell, and the presence of a discontinuity of elevation in the cell. To avoid frequent erroneous classification, the first two criteria are evaluated only if the number of points in the cell is large enough. In practice, a minimum of five points per cell is used. Once individual cells are classified, they are grouped into regions and sent to the local map maintainer.

Figure 7 shows the operation of the perception module in a typical outdoor scene. Figure 7(a) shows a video image of the scene and Figure 7(b) shows the corresponding range image used for evaluating terrain traversability. Figure 7(c) shows the elevation map obtained by converting the range pixels to a Cartesian coordinate system in which z is approximately the vertical direction with respect to the ground plane. The maximum elevation with respect to the reference plane is one meter in this example. Figure 7(d) shows the result of the traversability evaluation. In this display, the traversable parts of the map are set to 0, the untraversable parts are set to 1. The set of bushes and rocks on the left side of the scene are correctly identified as untraversable. The classification of Figure 7(d) is converted to a list of obstacle patches and sent to the local map manager.

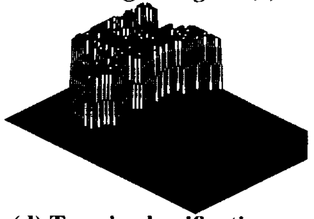


(a) A section of terrain from the path of Figure 2.



(c) Elevation map from the range image of (a).

(b) Range image of the terrain shown in (a).



(d) Terrain classification on the map of (c).

Figure 7: Example of range image processing.

This range image processing algorithm has several important properties. First, it does not build a complete, high-resolution map of the terrain, which would require interpolating between data points as in [7], an expensive operation. Instead, the algorithm evaluates only the terrain for which there is data. Second, the algorithm processes each image individually without explicitly merging terrain data from consecutive images. Instead, it relegates the task of maintaining a local map of untraversable regions to a separate local map module. The importance of this is that the local map module deals only with a few data items, the cells classified as untraversable, instead of with raw terrain data. As a result, maintaining the local map is simpler and more efficient. Because of these two features, range image processing is very fast, typically on the order of 200ms on a conventional Sparc II workstation. The main limitation is the 2 Hz acquisition rate of the sensor, not the processing time.

It is clear the range image processing module may miss untraversable regions of the terrain because the terrain is evaluated only where data is present in the image and because the data may be too sparse to provide complete coverage of the terrain at long range. However, because of the processing speed, a region that is missed in a given image will become visible in subsequent images quickly enough for the vehicle to take appropriate action. Although this problem effectively reduces the maximum detection range of the perception system, we argue that the other possible solutions would reduce the maximum range even further and would introduce additional problems. The most obvious solution is to merge data from a few images before committing to a terrain classification. This solution effectively reduces the maximum detection range because the system has to wait until enough overlapping images are taken before a terrain region is evaluated. In addition, merging images is in itself a difficult problem because it requires precise knowledge of the transformation between images. In particular, even a small error in rotation angles between two images may introduce enough discrepancy between the corresponding elevation terrain maps to create artificial obstacles at the interface between the two maps. (We refer the reader to [6] for a more quantitative description of this problem.) Therefore, unless the vehicle and position estimation systems are designed to produce very accurate pose estimates, it is preferable to not merge images explicitly and to rely on fast processing to compensate for the sparsity of the data.

5 Local Map Management

The purpose of the local map module is to maintain a list of the untraversable cells in a region around the vehicle. In the current system, the local map module is a general purpose module called Ganesha, developed by Dirk Langer [5]. In this system, the active map extends from 0 to 20 meters in front of the vehicle and 10 meters on both sides. This module is general purpose in that it can take input from an arbitrary number of sensor modules and it does not have any knowledge of the algorithms used in the sensor processing modules.

The core of Ganesha is a single loop (Figure 8) in which the module first gets obstacle cells from the perception modules, and then places them in the local map using the position of the vehicle at the time the sensor was processed. The sensing position has to be used in this last step because of the latency between the time a new image is taken, and the time the corresponding cells are received by the map module, typically on the order of 600ms. At the end of each loop, the current position of the vehicle is read and the coordinates of all the cells in the map with respect to the vehicle are recomputed. Cells that fall outside the bounds of the active region are discarded from the map. Finally, Ganesha sends the list of currently active cells in its map to the planning system whenever the information is requested. Because the map module deals only with a small number of terrain cells instead of with a complete model, the map update is rapid. In practice, the update rate can be as fast as 50 ms on a SparcII workstation. Because of the fast update rate, this approach is very effective in maintaining an up-to-date local map at all times. One last advantage of Ganesha's design is that it does not need to know the details of the sensing part of the system because it uses only information from early terrain classification. In fact, the only sensor-specific information known to the map module is the sensor's field of view which is used for checking for consistency of terrain cells between images as described below.

A different design of the local map module would be to maintain a much larger map with more information than just a list of terrain cells which would theoretically allow the navigation system to use data recorded from earlier images. There are two problems with this approach, however. First, the local map module is now forced to maintain a much larger amount of data, most of which is never used, introducing additional delays in the system. Second, errors in vehicle position accumulate to a point at which most of the map becomes useless. These two problems offset the occasional gain in additional information in the map.

In this design of the navigation system, the local map and planning modules do not have access to the original sensor data and therefore cannot correct possible errors in the output of the perception. In particular, a region which is mistakenly classified as traversable will never be reclassified because the local map module cannot go back to the original data to verify the status of the region. It is therefore important to use conservative values for the detection parameters in order to ensure that all the untraversable regions of the terrain are classified as such. The drawback of this approach is that the perception module may generate terrain regions which are incorrectly classified. For example, this may occur because of noise in the image or because of an erroneous reading of vehicle pose. Because the perception processes images individually without explicitly building maps, it cannot detect that this erroneous classification is inconsistent with previous observations. This problem is solved by the map maintainer which does maintain a history of the observations. Specifically, an untraversable map cell which is not consistent across images is discarded from the local map if it is not reported by the perception module as untraversable in the next overlapping images. Because the terrain classification is fast compared to the speed of the vehicle, many overlapping images are taken during a relatively short interval of distance travelled. As a result, an erroneous cell is deleted before the vehicle starts altering its path significantly to avoid it.

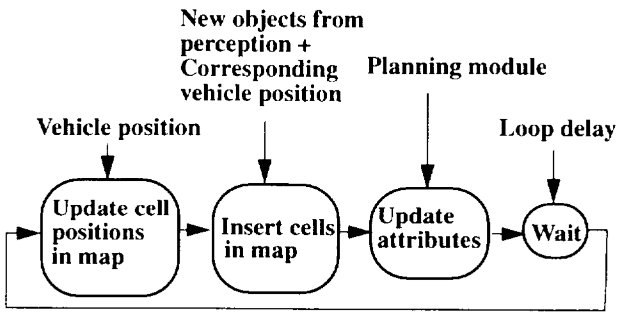
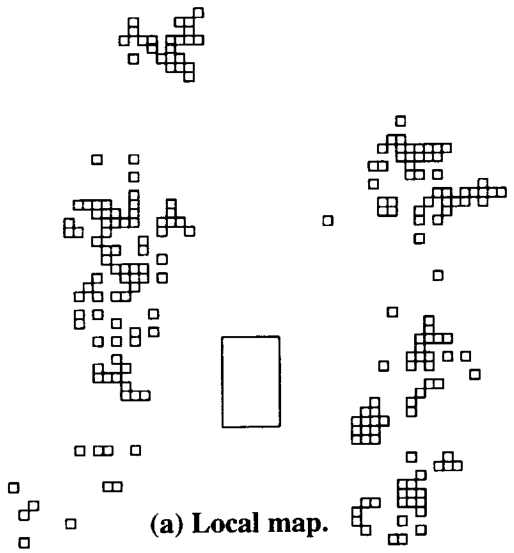

New objects from perception + Corresponding vehicle position
Planning module
Vehicle position
Loop delay
Update cell positions in map
Insert cells in map
Update attributes
Wait

Figure 8: Local map loop.

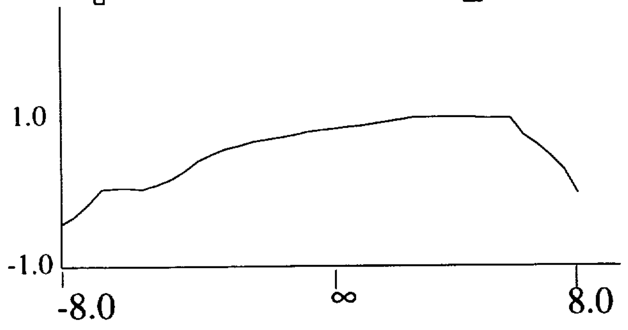
6 Path Planning

The last piece of the system is a trajectory planner which generates commanded steering radius and velocity with a high update rate. The trajectory planner, developed by Julio Rosenblatt [12][13], is composed of several modules. A set of two behaviors generates votes for every possible arc at the current vehicle position. An obstacle avoidance behavior computes the votes based on the distribution of untraversable terrain cells around the vehicle as reported by the local map module. Arcs that steer the vehicle away from the untraversable regions are given a high vote, while arcs that would cause the vehicle to travel through a forbidden region are given low votes. A second behavior gives higher votes to arcs that steer the vehicle toward intermediate goal points. This second behavior ensures that the overall path of the vehicle follows the desired global trajectory. The last module of the trajectory planner is an arbitrator which combines the votes from the two behaviors and sends the arc with the highest weight to the vehicle controller. Although we describe the architecture for trajectory planning strictly in the context of rugged terrain navigation, the architecture is very general in that it can accommodate a variety of behaviors, it is sensor-independent, and it can implement different strategies for combining weights.

Figure 9 illustrates the operation of the arc generation system. Figure 9(a) shows a display of the local map in the vicinity of the vehicle. The untraversable regions are displayed as before as squares corresponding to 40cm by 40cm terrain patches. Figure 9(b) shows the distribution of votes computed from this local map. The votes are between -1.0 and 1.0. The votes are computed for a list of 39 arcs with turning radii ranging from -8 to +8 meters.



(a) Local map.



(b) Corresponding distribution of votes.

Figure 9: (a) Example of local traversability map; (b) Distribution of votes in this example. The votes between -1.0 (forbidden arc) and 1.0 (clear arc) are computed for 39 arcs with radii between -8 and 8 meters.

The computation of the vote for a particular arc is controlled by three parameters: a maximum and minimum collision distance, and a near miss factor. These parameters are used as follows: Any arc for which the vehicle would collide with an obstacle cell at a distance less than the minimum distance is assigned a vote of -1.0; an arc which does not collide with an obstacle at a distance less than the maximum distance is assigned a vote of 1.0; and any arc which intersect an obstacle cell at an intermediate distance is assigned a negative vote weighted by the distance so that the vote increase as the collision occurs further along the arc. Finally, the near miss factor is used for penalizing the arcs which does not have any direct collisions but which pass close to obstacle cells. The votes decrease as the obstacle cells are closer to the arc.

This algorithm realizes a good compromise between the need to avoid obstacle regions, the need handle near-misses when an arc does not collide with an obstacle in order to take into account the uncertainty in the control system, and the need for limiting the lookahead distance of the planner in order to avoid situations in which the vehicle would be blocked by obstacles that are very far away and therefore do not pose any threat.

Because the trajectory planner generates only local arcs based on compact local information, the obstacle cells, it has a very high update and allows for rapid correction of small errors due to system delays or isolated perception errors. This is in contrast to the trajectory planner alternative in which a sequence of arcs is planned ahead instead of a single steering direction. In this case, trajectory planning is considerably slower and therefore introduces significant latency in the navigation system. A side-effect is that the system cannot recover from an error in the terrain map until it has already started executing a significant portion of the path through this map. This can be avoided by using more precise map building algorithms, but only at the cost of additional latency in the system. We refer the reader to [6] and [3] for a more precise description of the performance and limitations of this type of approach.

7 Conclusion

In summary, early evaluation of terrain traversability allows us to achieve continuous motion at moderate speeds by: reducing the amount of computation required by the perception system; simplifying local map management and path planning; hiding the details of sensing from all the modules except perception; and avoiding the problems caused by merging multiple terrain maps using inaccurate position estimates. The drawback of this approach is that an error in the perception system cannot be corrected later in the system because only the perception module has access to the sensor data. This problem is eliminated by using a fast reactive path planner and a simple perception algorithm with fast cycle time relative to vehicle speed, both of which allow the system to correct quickly for occasional perception errors.

While appropriate in many instances, this approach is not suited for all vehicles. In particular, we have made the assumption that the vehicle can safely negotiate terrain variations which are detectable far enough in advance that the vehicle is able to modify its path appropriately. For example, this vehicle at these speeds can tolerate terrain discontinuities of 20cm. With a range resolution of 7cm and an angular accuracy of 0.5°, such a discontinuity can be detected in time to avoid it with an arc of radius less than the minimum turning radius of 7.5 m, assuming a 2Hz image acquisition rate and an additional 0.5 seconds latency in the system. Sensor acquisition rate and resolution are the two numbers that set hard limits on the speed.

We have described the navigation system as a distributed system composed of three modules. Recently, we have improved our approach by merging all three modules into a single integrated modules. The integrated modules processes range images one scanline at a time, extracting obstacle regions, and maintaining its own local map internally. At regular interval, the module evaluates votes for a fixed set of arcs based on the current local map, much in the same way as the arc generation described in Section 6, and sends the votes to an arbiter which combines them with votes from external modules. This integrated approach allows for better performance by eliminating some of the latency due to the distributed nature of the system, and by ensuring that obstacle regions are reported as soon as they are detected by the perception processing.

Acknowledgments

The UGV project is supported by ARPA under contracts DACA76-89-C-0014 and DAAE07-90-C-R059 and by the National Science Foundation under NSF Contract BCS-9120655. Julio Rosenblatt is supported by a a Hughes Research Fellowship. The authors wish to thank Jay Gowdy, Bill Ross, Jim Moody, Jim Frazier and Mike Blackwell for their help in the cross-country navigation experiments.

References

[1]R. Brooks and A. Flynn. Fast, Cheap, and Out of Control: A Robot Invasion of the Solar System. *J. British Interplanetary Society*, 42(10):478-485, 1989.

[2]G. Giralt and L. Boissier. The French Planetary Rover VAP: Concept and Current Developments. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pp. 1391-1398, Raleigh, 1992.

[3]B. Brummit, R. Coulter, A. Stentz. Dynamic Trajectory PLanning for a Cross-Country Navigator. In *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[4]M. Hebert, E. Krotkov. 3D Measurements from Imaging Laser Radars. *Image and Vision Computing 10(3)*, April 1992.

[5]D. Langer and C. Thorpe. Sonar-Based Outdoor Vehicle Navigation and Collision Avoidance. In *Proc. IROS '92*. 1992.

[6]A. Kelly, T. Stentz, M. Hebert. Terrain Map Building for Fast Navigation on Rugged Outdoor Terrain. In *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[7]I.S. Kweon. *Modeling Rugged Terrain by Mobile Robots with Multiple Sensors*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, January, 1991.

[8]D. Langer, J.K. Rosenblatt, M. Hebert, "Off-Road Navigation", submitted to Special Issue of *IEEE Transactions on Robotics and Automation*, 1993.

[9]L. H. Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation. *International Journal of Computer Vision*, 8:1, 1992.

[10]E. Mettala. Reconnaissance, Surveillance and Target Acquisition Research for the Unmanned Ground Vehicle Program. In *Proc. Image Understanding Workshop*, Washington, D.C., 1993.

[11]K. Olin, D.Y. Tseng. "Autonomous Cross-Country Navigation". *IEEE Expert*, 6(4), August 1991.

[12]D.W. Payton, J.K. Rosenblatt, D.M. Keirsey. Plan Guided Reaction. *IEEE Transactions on Systems Man and Cybernetics*, 20(6), pp. 1370-1382, 1990.

[13]J.K. Rosenblatt and D.W. Payton. A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control. in *Proc. of the IEEE/INNS International Joint Conference on Neural Networks*, Washington DC, vol. 2, pp. 317-324, June 1989

6062

AIAA-94-1219-CP

THE ROAD PLAN MODEL - INFORMATION MODEL FOR PLANNING ROAD BUILDING ACTIVITIES

N94-30567

S11-63
301664
p. 11

Rafaela K. Azinhal
UNINOVA - Centro de Robótica Inteligente
Quinta da Torre, 2825 Monte da Caparica, Portugal

Fernando Moura-Pires
FCT/UNL - Dep. de Informática
Quinta da Torre, 2825 Monte da Caparica, Portugal

Abstract

The general building contractor is presented with an information model as an approach for deriving a high-level work plan of construction activities applied to road building. Road construction activities are represented in a Road Plan Model (RPM), which is modelled in the ISO standard STEP/ EXPRESS and adopts various concepts from the GARM notation. The integration with the preceding road design stage and the succeeding phase of resource scheduling is discussed within the framework of a Road Construction Model. Construction knowledge is applied to the road design and the terrain model of the surrounding road infrastructure for the instantiation of the RPM. Issues regarding the implementation of a road planner application supporting the RPM are discussed.

Introduction

The work presented in this paper is being done within the ESPRIT III project 6660 - *RoadRobot - Operator Assisted Mobile Road Robot for Heavy Duty Civil Engineering Applications*. The project is partially funded by the European Commission under the ESPRIT R&D programme and involves seven partners in five european countries, ranging from research and technology organizations, a manufacturing company as end producer and a building contractor as end user.

The objectives of this project are to adapt a generic control architecture to the requirements of the building industry and to build up and integrate components needed for automated out-door construction purposes. The operation of the developed subsystems and control strategies will be demonstrated under real conditions by the integration of two autonomous prototypes of the road building application: a road paving machine and an excavator.

The research institute Uninova is responsible for the development of the central site controller, which will integrate the working cells into a CIM environment, including functions of planning, scheduling, cost calculation, production and manufacturing supervision. Large amounts of information are generated and consumed during the various phases of a project life cycle. Sharing and maintaining these project data among multiple disciplines and throughout a project life cycle is a complex and difficult task. The project data needs to be stored, retrieved, manipulated and updated by many participants, each with his own view of the information. This leads to a step-by-step integration strategy, in which the several stages are carefully rationalized, automated and subsequently inserted in the global system. For a description of the multi-agent architecture proposed for the site controller, see [8]. This architecture is based on an object-oriented concept for modelling the product information as well as the processes, and is intended to link CAD systems, relational database, knowledge-based systems and other conventional application software.

The STEP standard

One of the problems of all CIM systems concerns the representation of the information to be accessed by the different agents of the (road) construction process. This implies the definition of common models for shared concepts in order to support an effective exchange of information. As the project favours the ideas of international standardisation works, we have considered the use of the ISO 10303 standard, the so-called STEP (STandard for the Exchange of Product model data) [5], to model the information inside the CIM system.

The STEP standard includes a formal information modelling language, called EXPRESS [4] used to specify the objects belonging to a universe of discourse, the information units pertaining to those objects and the constraints on those objects. All tools inside our site controller will model the information according to this formalism and be able to access instances of EXPRESS entities. This implies the development of STEP translators from supplier-specific file formats into EXPRESS entities which are then stored in the site controller's common information system.

The physical implementation of the information structure in a database will be based on a CIM architecture similar to the one presented in the ESPRIT II project IMPPACT (Integrated Manufacturing of Products and Processes using Advanced Computer Technologies) [3] - Chapter 2.4), and which was partially implemented by our research group within the European BRITE/ EURAM project CIMTOFI [10].

The GARM notation

The General AEC Reference Model [2], developed by W. Gielingh, describes the product model through so-called Product Definition Units (PDU). GARM is part of the draft proposal of the ISO standard STEP. The current version of the GARM concentrates on the requirement and design stages of the product life cycle.

Basically, PDUs describe the objects (or parts of an object) that have to be handled. A PDU appears in different stages during its life cycle: required stage, design stage, planning stage, production stage, etc. Only the first two life cycle stages are worked out. A PDU in the 'as-required' stage is called Functional Unit (FU). A PDU in the 'as-designed' stage is called Technical Solution (TS). These two stages are used for decomposition during the design of a PDU.

GARM is based on the FU-TS decomposition. This construct expresses the fact that a top-down design process is ruled by the divide-to-conquer principle. Searching a TS for a set of requirements collected in a FU is done by breaking the TS up into lower order FUs, i.e. by dividing the problem into a number of smaller design problems.

This principle can be visualized by means of a so-called Hamburger diagram (Fig. 1). Such a diagram represents the product model as a hierarchical tree whose nodes consist of two semi-circles. The upper side symbolizes a FU and the lower side the selected TS. Decomposition levels may coincide with responsibilities, disciplines, contractor/ subcontractor/ manufacturer relationships, etc.

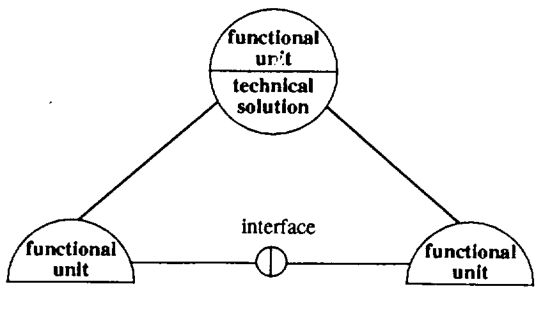


Fig. 1: Hamburger diagram: decomposition tree

Besides these vertical relations, which structure the FUs and TSs into a hierarchical tree, the various components at a FU-level may be related to each other (horizontal relations). GARM relates the FUs mutually by means of a network. These relations are called interfaces.

In the next chapter, we present a model that describes all the properties of a family of roads during the design process. Such a model is called a *product type model*. During the design process, a product model for a specific road is generated by choosing those properties from the product type model that are needed to fulfil the specific requirements of that road.

The Road Model Kernel

During the analysis of the state-of-the-art in integration of new technologies into building industry, it was evident that most established developments concentrate on computer-aided drafting. Here, several CAD packages from different software suppliers have been identified. Nevertheless, it also became obvious that one big problem associated to the rapid increase of specific CAD-software programs is the ability to exchange information between each other, not to mention with programs with other purposes during the life cycle of a product.

The Dutch Ministry of Transport, Public Works and Water Management, in conjunction with TNO Building and Construction Research, has seen the need to lay a new foundation of a new standard for road development. This has led to the development of the so-called "Road Model Kernel" [11], a product description of the road in the design stage based on the ISO/STEP standard. A STEP translator was developed which allows the exchange of the MOSS file format with the RMK without loss of information.

The RMK was developed using GARM's methods, and therefore describes the road in terms of FUs and TSs. However, the PDUs of the RMK are not real-world objects (or parts of objects) which can be obtained independently through construction processes and jointly form the 'as-built' road. Instead, they have been defined to reflect the viewpoint of the road designer as he conquers the complex problem of designing a road.

In several internal models used by road modelling packages, one often encounters two layers of decomposition: firstly a longitudinal decomposition and secondly a transversal decomposition. This longitudinal decomposition is often split into a longitudinal decomposition to describe the horizontal alignment and a longitudinal decomposition to describe the vertical alignment.

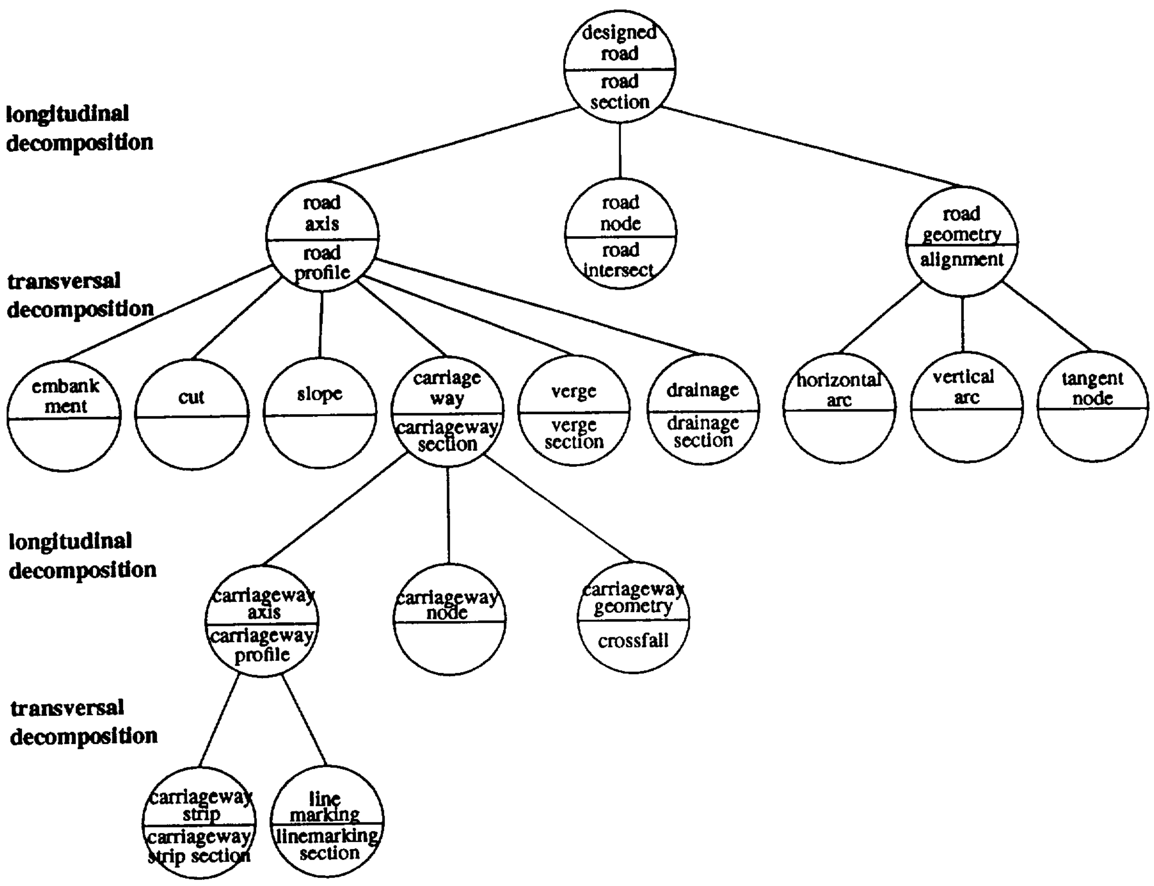


Fig. 2: The Road Model Kernel

The principle to decompose alternately and on hierarchically different levels into longitudinal and transversal seems to fit properly with the experience of the road designer, and was adopted by the RMK.

Road-axes and road-nodes constitute the framework to describe the structure of the roads and their connectivity: the **topology**. However, this description does not incorporate sufficient information to extract the accurate shape of the road. This is done by adding the **geometry** to the road-axis as a separate entity.

The FU road-geometry shapes one or more road-axes which will assemble a continuous chain. The road-geometry will make demands on the progression of curvature, horizontal as well as vertical. Alignment is the TS which can be selected for the FU road-geometry. Alignment decomposes into two interconnected networks (chains) which describe separately the horizontal and vertical alignment.

For the geometrical representation of the road, a specific type of coordinate system must be chosen for the RMK. Because of its simplicity and flexibility, the RMK uses a floating around the z-axis rotating s-t-z coordinate system, which is related to the horizontal alignment curve.

The s-axis maps one-to-one on this curve (longitudinal direction) and is embedded in the x-y plane. The t-axis is orthogonal (perpendicular) to the s-axis (transversal direction) and is also embedded in the x-y plane. The z-axis is equivalent to the z-axis of the fixed x-y-z coordinate system.

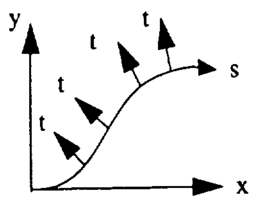


Fig. 3: Coordinate system s-t-z vs x-y-z

The horizontal and vertical alignment description in the RMK defines the curvature functions along the longitudinal (s-axis) direction of the road.

The alignment incorporates a chain of arcs interconnected by tangent nodes. Arcs may specify no curvature (straight), one (circular curve) or two curvatures to denote start and end magnitude (linear transition).

The horizontal and vertical alignments are defined at a high level, dragging all lower level entities to follow automatically this primary shape. However, the influence of a crossfall is dedicated to a specific transversal function. Therefore, a geometry entity should be imposed only to that specific transversal function (carriageway geometry, slope geometry, ...). The TS crossfall decomposes subsequently into a collection of tangent nodes containing the magnitude of a specific gradient.

The Road Construction Model

As presented by J. Everett in his paper [1], construction and manufacturing exhibit fundamental differences in where the interface or transition occurs between product design and process design or fabrication. In repetitive manufacturing operations, the product-process design team controls product and processes all the way down. However, in construction, there is little overlap between product design and process design or fabrication. Architect/ engineers control product design but do not get involved in the building process other than to inspect the finished work for conformance to design specifications. Constructors control the fabrication process design but generally have little or no input into product design. A distinct separation exists between the product designers and or architect/ engineers, and the process designers or craft workers. In construction, the product designers and process designers are almost always separate organizations with different objectives.

This is specially true for heavy-duty civil engineering applications, like road construction, where the gap between the lower limit of design detail and the upper limit of machine technology is substantial, as very few practical examples of construction robotics or highly automated machines have been developed.

As seen above, the Road Model Kernel represents the road design without any detail about the processes used to build the road. Until the product design and process design can be integrated by closing the gap between design and machine technology, we propose to use a step-by-step integration strategy which reflects the current way of work.

During contacts with the entity responsible for the construction of highways in Portugal, Brisa, the following agents were identified and a description of their roles in the construction process is given below:

- Based on the user's needs, the **construction owner** Brisa defines the requirements of the road to be built and delivers these to the design team.
- As the national **authority** for the design of highways, Brisa distributes the *design regulations* to the design team, which include for instance minimum values for the radius of curves depending on the requested speed, ways of calculating the earth volumes, norms about the composition and thickness of the paving layers depending on the soil resistance, etc.
- The **design team** returns to the construction owner a set of documents (descriptive memory) as result of several design activities, such as (a) geometric drawings, (b) earthworks based on geological/ geotechnical studies, and (c) paving layer composition of the road sections.
- As the national **authority** for the construction of highways, Brisa distributes *general technical norms* to the general contractor about the construction of highways, for instance about the material to use in earth-filling, classification of the soil based on specific attributes and their application, the notion that the vertical alignment as defined in the design drawings refers to the surface course of the road or to the compacted base platform as well as transversally to the line limiting the carriageway and the left verge, the proceedings for the quality control of the earthworks and the paving, etc.
- The **general contractor** hired receives the design documents from the construction owner as well as a contract specification book. The contract document specifies additional *construction requirements* to the general technical norms. Based on these, the general contractor plans how the road is to be built in order to maintain the requested deadlines and costs, requisitions the resources and carries out the site production, eventually by hiring sub-contractors.
- **Sub-contractors** perform tasks and produce the products or components for the construction, for instance a sub-contractor is hired to build the bridge, another is hired to do all earthworks, etc.
- **Machinery lending firms** provide equipment to the site.
- **Suppliers/ distributors** supply and distribute material for the site facility, such as the asphalt plant supplies the asphalt mixture for the road paver, and gas

stations supply petrol for the machinery, etc.

As the RoadRobot project embarks all phases of road building from design to production, the RMK will be used as the 'as-designed' model of the road. For later processes like planning and production, and for the modelling of resources and activities, new modelling constructs have to be found and added to the previously presented GARM model. The Road Construction Model will be based on B. Luiten and F. Tolman's "Building Product Model (BPM)" [6], and will be used in our work for the integration of design and construction knowledge and information.

For the Road Construction Model, the following stages have been identified (see Fig. 4):

- the design stage, where the product road is described by the road designer in terms of its geometric requirements -> Road Model Kernel,
- the planning stage, where the activities are identified by the general contractor as constant road sections to which they apply -> Road Plan Model,
- the scheduling stage, where to each activity identified at the previous stage the resources to realize them are assigned by the general contractor, in order to optimize time and costs -> Road Schedule Model,
- the construction stage, where the tasks are effectively issued to the working cells and their execution monitored, resulting in a built road which will be inspected relatively to its requirements.

In a building project, three main groups of entities can be modelled: the Product, the Activities and the Resources. The information about these entities can be

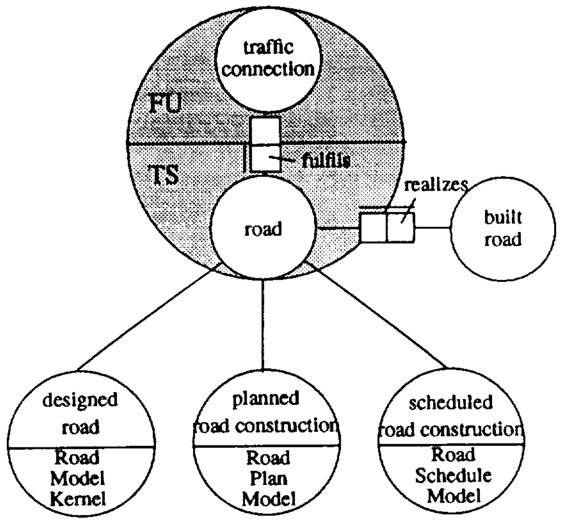


Fig. 4: The Road Construction Mode

modelled in respectively a Product Definition Unit (PDU), an Activity Definition Unit (ADU) and a Resource Definition Unit (RDU). GARM has worked out concepts for PDU which are also suitable for ADU and RDU. To reuse modelling constructs and to avoid redundancy, common properties for PDU, ADU and RDU are modelled in a new entity called Construction Definition Unit (CDU).

The relations between Product (PDU), Activities (ADU) and Resources (RDU) can be graphically modelled in NIAM (Fig. 5).

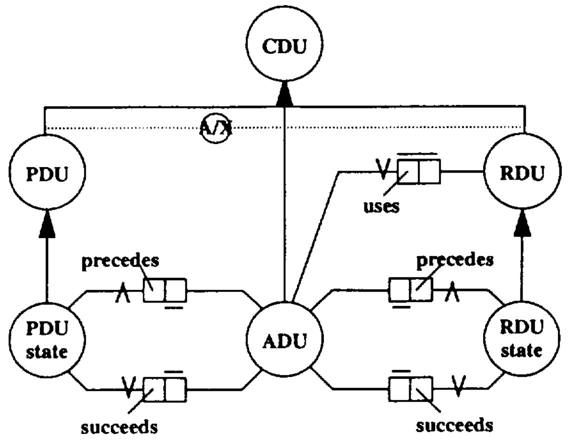


Fig. 5: NIAM diagram of the Building Product

Examples of PDUs are: the product itself, parts of the product or features. For ADUs one can think of all the processes during the project: management, design, planning, and production processes. RDUs are resources used by ADUs, like manpower, equipment and raw materials.

For a PDU the main characteristics are 'shape' and 'material'. Other characteristics can be derived from the main characteristics. For an ADU the main characteristics are 'time constraints', e.g. 'must be performed before or after', 'can be performed independently of'. For a RDU all characteristics have something to do with 'money', e.g. 'application costs', 'acquisition costs', 'remainder value'. When the ADUs are related to RDUs, absolute time can be derived, e.g. 'starting time, 'ending time' and 'duration'.

Examples of states are 'as designed', 'as planned' and 'as built'. In general, an ADU is preceded by a PDU state and succeeded by a new PDU state. An ADU always uses one or more RDUs. It is possible that this ADU also changes the state of the RDU.

As an example of the use of the Road Construction

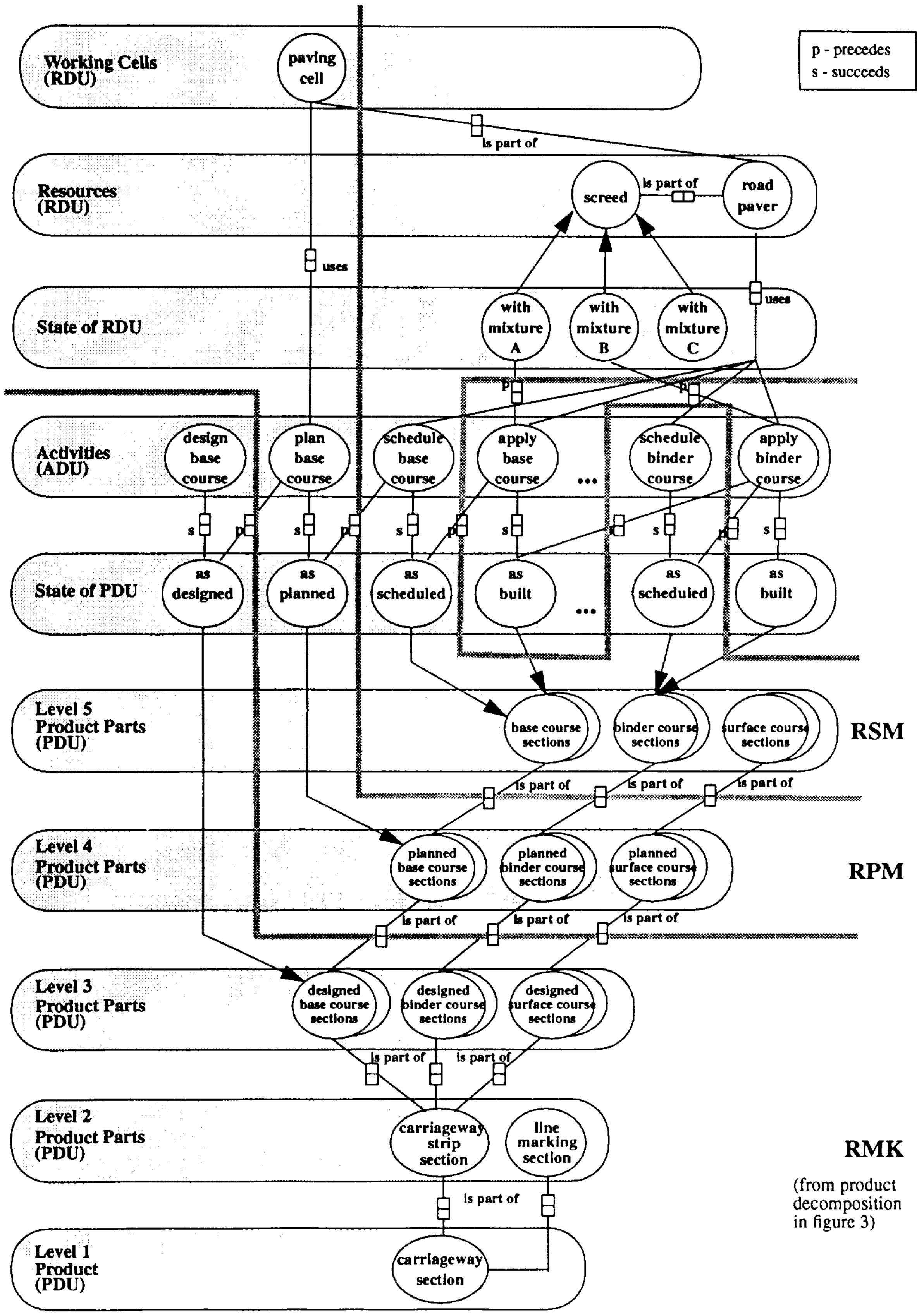


Fig. 6: NIAM diagram of the paving activity of a carriageway section

Model, the paving activity of a three course carriageway section is partially worked out. The pavement consists of three asphalt courses which are sequentially applied over the preceding course.

In Fig. 6, this activity is modelled in a NIAM diagram using the concepts of the Road Construction Model. At the three bottom levels of the diagram, the PDU decomposition is the one followed by the road designer as identified in the Road Model Kernel of Fig. 2. The fourth level models the PDUs identified by the general contractor when planning the paving process of the designed carriageway, as will be shown in the Road Plan Model at Fig. 7, taking into account only the restrictions imposed by the surrounding road infrastructure. The fifth level models the PDU decomposition followed for scheduling the planned paving process (Road Schedule Model), considering the time constraints and the resources available at the building site. Each of the Activities 'design', 'plan', 'schedule' and 'apply' models the transition of one Model to the succeeding Model of the Road Construction Model, changing the state of the PDU from 'as required' to 'as designed', to 'as planned', to 'as scheduled' and finally to 'as built', respectively.

The Road Plan Model

In the present work, a "Road Plan Model" will be proposed, which describes the road in the 'as-planned' stage. In the same way as the RMK, the RPM represents the viewpoint of the general contractor when he takes the complete contract document delivered by the construction owner and creates the high-level work plan of construction activities.

The purpose at the planning stage is to identify the road sections which require different types of construction activities, and their dependencies. Each of these activities can be visualized as being executed by a working cell composed of a set of resources which work jointly to realize that activity. These working cells are logical entities which will be instantiated during the scheduling stage with the necessary quantity of resources (machines and humans) in order to maintain deadlines and budgets.

During contacts with several building contractors, the following high-level construction activities were identified, which are presented graphically in the GARM tree of Fig. 7. This tree forms the basis of the so-called "Road Plan Model".

A problem which was encountered in this stage of development, was the selection of proper names for all entities and objects essential to construct the data model. To provide some pattern to it, the next schema was used: The FU of the RPM identifies the *activity* to execute over a specific road section. The TS specifies the *working cell* which satisfies that requirement.

For example, suppose a specific road section passes over a valley or a river. The FU describes this road section with *bridge construction activity*, indicating that the TS to obtain such a road section is the construction of a bridge by a *bridge construction cell*.

Another example relates to the land *clearing activity*, which is satisfied by the *clearing cell*. The selection of the equipment composing this particular working cell depends on the diameter of the vegetation and on the size of the area. However, these questions are answered only at the scheduling stage, as the selection of equipment is also affected by whether there are alternate uses for equipment as well as by time limits.

During the development of this model, a decision had to be taken concerning the depth of the RPM decomposition tree, i.e. the granularity of the working cells. As our purpose is to model the way of thinking of the general contractor while building the high-level work plan, the result is the one shown in Fig. 7. However, the adopted GARM concept, which separates FUs and TSs, allows for more details to be added at the end (leaves) of the model. Specifically, this is done when planning and scheduling the resources inside a working cell.

The granularity of the lower-order activities in the RPM (level of the leaves) defines the functionality of the working cells which can realize them and which will be allocated in the scheduling stage. In turn, each of the working cells must be able to plan and monitor the execution of each of its resources (machines and humans). The higher the functionality of the working cells is, the more complex is the management of their resources. Here, the same approach to the just described CIM system can be applied.

The vertical decomposition identifies road sections where the named activity is applicable and their decomposition into sub-sections for lower-order activities. The identification of each road section depends on the activity to perform over it, which in turn depends on the connection of the road design to the surrounding road infrastructure. Usually, the general contractor defines a road section by indicating the initial and final station, i.e. the s-ordinate in the s-t-z coordinate system of the road design.

The horizontal network structure describes the sequences and dependencies of the construction of each of the road sections, and therefore of the activities which

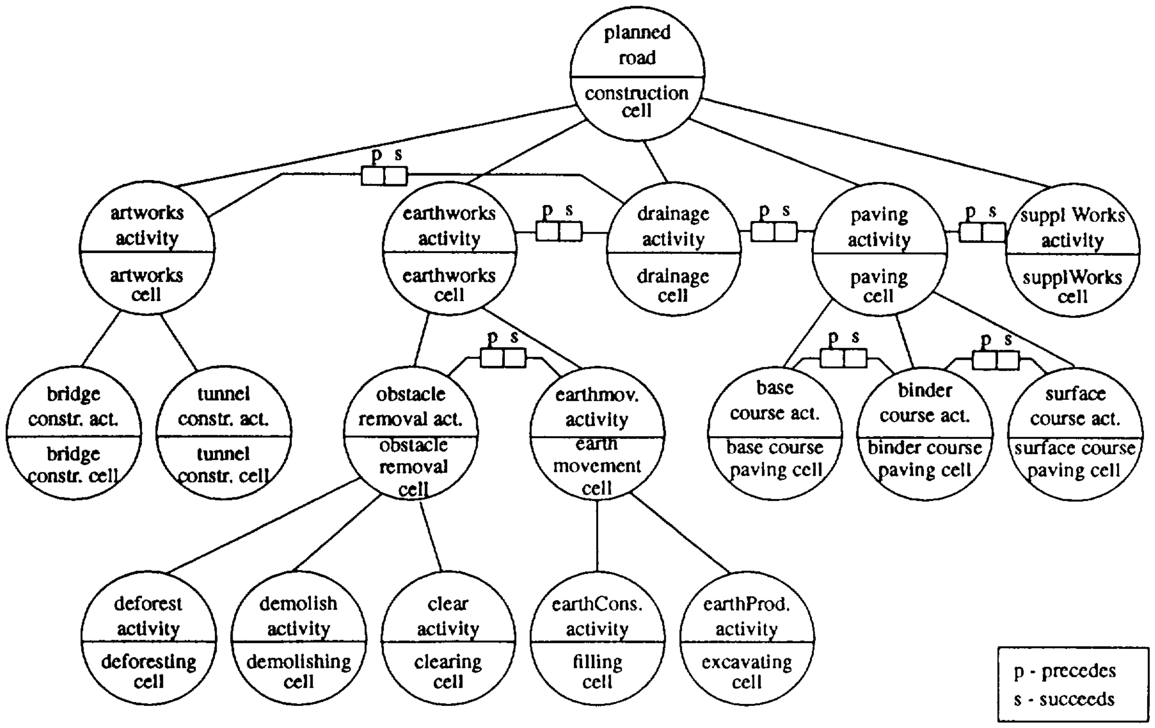


Fig. 7: The Road Plan Model

realize them (*precedes, succeeds*).

Over the same road section, the activities have a well-defined precedence. For instance, earthmoving is performed before drainage, and drainage is done before paving, the surface course is put on top of the binder course, the art works (bridges, tunnels) are done in parallel with the earthworks prior to paving.

Between different road sections, it is also possible to define precedency. For instance, paving a road section is an activity which is further decomposed into lower-order activities: apply base course, then binder course and finally surface course. However, the successive application of each of the courses does not have to be made over the total length of the road section. That is, the road section to be paved may be decomposed into sub-sections, which allow a different, even simultaneous application of the layers; for instance apply the base course over the initial sub-section, then apply the binder course over that same sub-section, while applying the base course over the second sub-section, etc.

This flexibility facilitates the scheduling of the activities, allowing for the optimization of the temporal allocation of the working cells to each of the planned activities. For example, if two paving cells are available at the building site, their simultaneous use makes it possible to optimize the execution time of the global activity of paving a road section. Once activities are attached to product parts and resources to activities, production time and costs can be predicted.

Implementation issues

Within the RoadRobot project, this work will result in the implementation of a "Computer-Aided Planner". Taking the instantiated RMK, the construction specifications and some terrain model, this expert system will aid the general contractor in creating an instance of the RPM by specifying the road sections and the construction activities which have to be realized over them.

The proposed situation for the planning process is

Extract of the EXPRESS description of the Road Plan Model

```
SCHEMA RoadPlanModel;

...

ENTITY RoadPlan;
     plannedRoadActivity: ConstructionCell;
END_ENTITY;

ENTITY ConstructionCell;
     artWorkActivity:    LIST [1:?] of ArtWorkActivity;
     earthWorkActivity: LIST [1:?] of EarthWorkActivity;
     drainageActivit:y  LIST [1:?] of DrainageActivity;
     pavingActivity:    LIST [1:?] of PavingActivity;
     supplWorkActivity: LIST [1:?] of SupplWorkActivity;
END_ENTITY;

...

---------------------- Paving -------------------------

ENTITY PavingActivity;
     requiredCell:       PavingCell;
     precedes:           SupplWorkActivity;
     section:            Section;
END_ENTITY;

ENTITY PavingCell;
     baseActivity:       LIST [1:?] of BaseCourseActivity;
     binderActivity:     LIST [1:?] of BinderCourseActivity;
     surfaceActivity:    LIST [1:?] of SurfaceCourseActivity;
END_ENTITY;

ENTITY BaseCourseActivity;
     requiredCell:       BaseCoursePavingCell;
     precedes:           BinderCourseActivity;
     section:            Section;
     qt:                 Ton;
END_ENTITY;

...

END_SCHEMA;
```

described by the IDEF0 diagram [9] on Fig. 8.

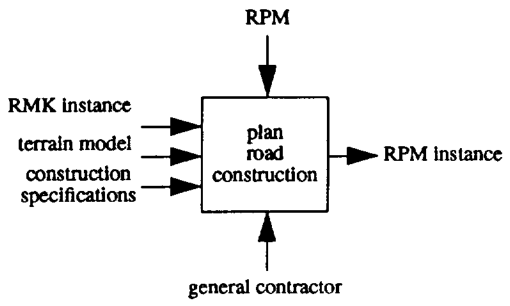


Fig. 8: IDEF0 representation of the planning stage

Construction specifications

The translation of design information into process planning information is a translation process in which construction knowledge is applied to the design information. This knowledge can be classified into three categories according to the scope of the statements:

- *general building knowledge*
  knowledge applicable to every building product and project.

- *product type specific building knowledge*
  knowledge applicable to specific product types, e.g. roads.

- *product specific building knowledge*
  knowledge applicable to specific products of a certain supplier, e.g. highways.

- *project specific building knowledge*
  knowledge applicable to a specific building project..

The first category of knowledge refers to general construction knowledge. An example is the selection of the activity depending of the type of vegetation of the terrain at the building site. If there are trees, then there has to be an activity which cuts them off; if there is a building, then it must be demolished, be it for the construction of a road or of a building.

The following two categories of knowledge are usually available in regulations. For example, the width of the paving courses is determined by the type of soil under the road and the traffic which should be supported by the road. The first variable is given by the terrain model, the second one is specified in the requirements of the road design.

The last category of knowledge is specific to a particular construction project. The construction owner may specify that, during the earthworks, the soil of the platform is to be made constant, even when this means getting soil of the required resistance from a distant earth deposit, as it is impractical to vary the thickness of the asphalt courses during the paving process. Another example is the specification of the quality control points.

Terrain model

One problem of the RPM is the integration of the road design with the surrounding terrain model. Two viewpoints over the terrain are relevant: the geotechnical and the topographical.

The geotechnical model describes the road corridor in terms of the geological characteristics of the underground soil: sand, rock, underground water rivers, etc. This model is important for the planning of construction activities and at the scheduling stage for the selection of resources to apply during a specific construction activity: use a motorscraper for earthmoving sand, but an excavator for earthmoving clay.

The topographical model describes the surface of the road corridor, including the identification and location of natural and human-made obstacles: vegetation, rivers, buildings, etc. This model allows the general contractor to plan the way to deal with each of the obstacles, namely which construction activity to use: demolish a building, cut off trees, build a bridge over a river, etc.

Obviously, the functional modelling of the terrain in ISO/STEP is by itself an own project. Therefore, within the RoadRobot project, for the implementation of the site controller, an industry standard format will be selected for the digital terrain model (DTM), for instance the format TIN, which defines the surfaces by means of triangulated 3D facets.

Conclusion

This work suggests a Road Construction Model using concepts of GARM. For the decomposition of a PDU during its life-cycle, we chose to follow the construction process as much as possible, which resulted in the creation of several models, because such a decomposition supports all the aspect views without being far away from the mental world of the users in practice.

The first model dedicated to the design process, the Road Model Kernel, was developed by the TNO- Building and Construction Research institute and has already a working computer version.

The present paper presents a conceptual model for the planning process of the road construction as practised by the general contractor, the Road Plan Model.

Further work

To allow for the integration of design and construction with the developed Road Construction Model, models for activities and resources have to be worked out, similar to the product model, as well as the relations between these three entities. This includes detailing and implementing the Road Plan Model and the Road Schedule Model as was done with the Road Model Kernel.

References

[1] John G. Everett. Design-Fabrication Interface: Construction vs. Manufacturing. In R.L. Tucker G.H. Watson and J.K. Walters, editors, *Automation and Robotics in Construction X*, number ISBN 0 444 81523 6 in Proceedings of the 10th ISARC, pages 391–397, Houston, Texas, U.S.A, May 1993. Elsevier Science Publishers.

[2] W. Gielingh. General AEC Reference Model (GARM). In *Proceedings CIB W74+78*, Lund, Sweden, October 1988.

[3] W.F. Gielingh and A.K.Suhm, editors. *IMPPACT Reference Model*. Number ISBN 3-540-56150-1 and 0-387-56150-1 in Research Reports ESPRIT, Project 2165 IMPPACT Vol.1. Springer-Verlag, 1993. An Approach to Integrated Product and Process Modelling for Discrete Parts Manufacturing.

C 5.

[4] International Standards Organization. *Part 11: Descriptive Methods: EXPRESS Language Reference Manual*, ISO DIS 10303-11 edition.

[5] International Standards Organization. *STEP - Standard for the Exchange of Product Model Data*, ISO DIS 10303 edition.

[6] Bart Luiten and F.P. Tolman. Design for Construction (DFC) in the Building and Construction Industries. In Harry Wagter, editor, *Proceedings CIB W74+78*, number ISBN 0 444 89262 1 in Computer Integrated Construction, pages 137–144, Tokyo, Japan, September 1992. Elsevier Science Publishers.

[7] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, 1989.

[8] J.P. Pimentão et al. Architecture of the Site Controller for Road Building Applications. In P.A. MacConaill C. Kooij and J. Bastos, editors, *Realising CIM's Industrial Potential*, pages 358–367. IOS Press, Amsterdam, May 1993.

[9] Douglas T. Ross. Structured Analysis (SA): A Language for Communicating Ideas. In *IEEE Transactions on Software Engineering*, volume Vol.3, No.1, pages 16–33. IEEE, January 1977.

[10] M. Tavares et al. Tool Integration in CIM Environment - Looking for Standardisation. European BRITE/EURAM Project B-E 3653 - CIMTOFI.

[11] P. Willems. The Road Model Kernel. Technical Report TNO B-89-831(E), TNO - Building and Construction Research, January 1990.

→ MG
done CMG

AIAA-94-1220-CP

6063

N94-30568

Design Reuse Experience of Space and Hazardous Operations Robots

P. Graham O'Neil
Automation and Robotics Department
Lockheed Engineering and Sciences Company
2400 NASA Road 1, Houston TX 77058
oneil@aio.jsc.nasa.gov

5/2-63

[illegible]

p 6

January 12, 1994

Abstract

A comparison of design drivers for space and hazardous nuclear waste operating robots details similarities and differences in operations, performance and environmental parameters for these critical environments. The similarities are exploited to provide low risk system components based on reuse principles and design knowledge. Risk reduction techniques are used for bridging areas of significant differences. As an example, risk reduction of a new sensor design for nuclear environment operations is employed to provide upgradeable replacement units in a reusable architecture for significantly higher levels of radiation.

1 Introduction

Robotics operations in hazardous environments are attractive because they reduce exposure and risk to humans, perform reliably in hostile environments, and can be used to amplify human capabilities. The environments receiving the most attention for these applications have been underwater, outer space or on earth in areas where radioactive or hazardous materials pose threats to humans and their automated equipment. These environments possess some common characteristics, yet each is distinct in its engineering design challenges. As technology growth presents more economic alternatives, this list of working environments will grow. This paper presents a framework for cataloging reuse features, assessing benefits of the transfer from one environment to another and emphasizes the decisions made early in the life cycle, for optimal reuse.

Applications of design reuse and tailoring techniques for applications as diverse as laboratory automation of hazardous contaminants for the Contaminant Automation Analysis (CAA) Program to automated equipment used in hazardous waste tank operations, burial pit operations and mining extraction processes will be presented. An overview is presented in Table 1 for these applications and environments.

Typical design features driven by non-functional requirements that reuse knowledge or design details common to both space and hazardous waste operations equipment include the following:

- Safety
- Development risk reduction
- Manufacturing and production quality requirements
- Environment
- Human Machine Interface design for efficiency and safety
- Maintainability
- Reliability

A comparison of design features based on these requirements for laboratory analysis applications projected for environmental and space operations is given in Table 2.

A direct comparison of operational environments for the inspection tasks and light utility duty is given in Table 3 for the Special Purpose Dextrous Manipulator System being developed for the Space Station and the Light Duty Utility Arm for inspection tasks in the Hanford Single Shell Tanks.

2 Common Design Goals

For automated remediation operations in hazardous waste tanks, key operating parameters are driven by task/path planning and motion control. A high level view of the automation activities associated with task planning and execution of operators goals includes these activities:

- Direct robot to start task

Application	Nuclear	Space	Underwater
Lab Analysis	CAA	DART	—
Inspection	Hanford tanks	Space Station	Oil Rig and Cable Inspection
Soil Movement	INEL Pit 9	Lunar regolith processor	Dredging and Harbor cleanup
Assembly	Fuel Reprocessing	SS maintenance, FLO prep	Pump and Pipe placement

Table 1: Application and Environment Analogies

Design feature	DART/NASA	Contaminant Automation Analysis [DOE]
Packaging	Must assure containment	Integrated and self-contained
Workspace	Standard Lab module	Up to 8 standard modules in series
User Interface	Virtual Operator/Telepresence	High level user interface
Remote operator latency	> 3 seconds	1-3 seconds
Computer architecture	Real time UNIX	Real time Unix
Chemical Exposure	O, UV, high pH	acid fumes pH ↑ 14
Radiation [dose]	2 Rad/hr	250 Rad/hr
Radiation [lifetime]	10^5	10^8

Table 2: Laboratory Automation Comparison

- Search world model for access points
- Reason about tool selection for task
- Plan trajectory and workspace motion
- Present to operator for verification using the following interface channels:
  - Remote Viewing
  - Controller Inputs
  - Shared Control Authority
  - Graphics Display
  - Audio Feedback

Adapting the first 4 activities for the unique features of the Hanford single shell tank waste remediation system, the following extensions for Force Controlled Interactions with the environment include are developed:

- Minimize normal force on walls
- Surface tracking of solid waste
- Threshold force application for breaking salt formations and selection of appropriate tool for autonomous grinding or sucking tasks.
- Oscillation compensation for dextrous tool application

Similar tasks involving force controlled interactions with worksite and environment exist on planetary surfaces for resource utilization and in the underwater environment for harbor dredging, or oil well infrastructure development.

3 Reuse Processes

This section presents details on the use of scenario analysis and testbed development and utilization to enhance the reuse process. The following section presents details on the role of design knowledge transfer, interface specification, and trade study reuse.

Scenario Analysis

The scenarios analysis is a tool used for requirements development and analysis. It provides a framework for multi-discipline teams to describe events and flows within the system and perform contingency modeling and analysis. The process can be applied to an conceptual and architectural model to investigate requirements defects. Scenario analysis has been used in the following manner.

Figure 1 is a scenario outline used for deriving advanced development requirements for maintenance robotics for the First Lunar Outpost (FLO). When similar approaches were applied to buried waste retrieval tasks, the result included better definition of tether monitoring and management tasks, and the inclusion of trades for periodic decontamination and maintenance actions.

Simulation and Testbeds

Design processes for implementing these functions for robots operating in a variety of environments [undersea, in the field, or in space], can use similar simulations, and analysis tools, further increasing the potential for reuse of robotic design knowledge to field reliable systems with greatest design maturity and least development risk. Examples of reuse of these simulations and testbed facilities are:

Parameter	Space [SPDM]	Radioactive Waste [LDUA]
Temperature Range	-150 to +150 F	[-20 to 150 F]
Pressure Range	0 to 29.92	standard +/- 10 in Water
Particulate	Micrometeoroid	severe dust st
Chemical Exposure	O, UV	acid fumes pH ↑ 14
Radiation [dose]	250Rad/Sec	2,000 Rad/hr
Radiation [total]	10^5	10^8
Landing/Wind Loading	4.3 g	3.5g at 120 mph gust

Table 3: Operating Environment Characterization

1. Testbed facilities being developed at JSC have capabilities with broad application for other areas. Some of these features included in the testbed harnesses are remote operations, virtual reality interfaces, variable time delay loops, and coordinated multi-arm controllers. Other engineering test capabilities include instrumented dynamics testing facilities with useful payloads into the thousands of pounds.

2. Simulations of space environments can be developed form Earth based analogs if design features were embedded during development. For instance, one heavy equipment company has an analytic simulation of soil blade interactions with user specified inputs for soil characteristics and reduced values of gravity that would be suitable for lunar or Mars resource utilization advanced development studies.

3. Libraries of graphic kinematics for a wide range of robots, worksites, and operating environments are becoming available and with maturing engineering management direction should be critical in shortening the design cycle, minimizing design and schedule risk, and provide early access to the user community.

4 Reuse Strategies

This section presents an approach to applying the software engineering concept of Abstract Interface Specification to engineering design reuse and presents 2 examples of design knowledge transfer from diverse robotics fields to a design for inspection for the Hanford Single Shell Tanks.

Interface Specification

Based on experiences at software reuse, modifications were made to the approach that seemed the easiest to automate, the Abstract Interface Specification (AIS). The modifications included specification of technology maturity, remaining areas of risk, physical descriptions and resource requirements and reuse history.

This approach is also used in generating simulations based on reusable components. Briefly the AIS approach consists of specifying the required and provided services for each component and information about state constraints and exception conditions. Since files constructed with these attributes can be browsed with reuse software, the effort was minimal to setup and use. Figure 2 is an example of the extended abstract interface specification entries for this approach.

Entries in the specification are augmented by physical descriptions and operations notes with resource budgets where required. In particular slots are assigned for design knowledge and application reuse history of the following items:

- Materials
- software
- persistent design objects
- standard mechanical components
- standard trades

Once a library of specifications in this format is established, domain engineers can query or browse for suitability and closeness of application using a Knowledge Dictionary System approach. Establishing this library is one of the critical items in developing an engineering reuse process. This library and its tools for browsing provide the ability to use Commercial Off-The-Shelf (COTS) components with confidence, design robust systems with mature deissgn techniques, and include details for unique requirments based on modifications of closely approximated configuration items.

Design Knowledge Transfer

The use of standard trades studies and design knowledge transfer is illustrated by the following 2 case studies for the Hanford single shell tank Light Duty Utility Arm (LDUA) inspection system. Characterization of design features in general terms with parametrization for different operations environments.

1. A vertical positioning mast for contamination containment and housing of a robotic inspection arm is necessary for the Hanford single shell tank inspection task. Given the geometries of the specified delivery system, a multi-jointed mast is required to meet the volume and length specifications. Figure 3 shows the trade variables that were examined and the evaluation of the engineering team for each of the three major concepts. Each of the 5 evaluation variables,

   (a) Stiffness
   (b) Smooth external surface
   (c) Mast wall thickness
   (d) Actuation
   (e) Hinge design

   were chosen based on their contribution to top level requirements. Stiffness was derived from position accuracy requirements, smooth surface from contamination control and sealing requirements, and mast wall thickness from gross weight requirements. Design knowledge from experience in emplacement of masts for oil well drilling, marine structures, and simulation of multi-segment Space Station Remote Manipulator System (SS-RMS) boom assembly was used to complete the figures of merit for positive contribution and relative importance weightings.

2. The selection [shown in Figure 4] of a mast position sensor component was driven by requirements for position accuracy, robustness in field operations and cost. Characterization of the 4 design choices,

   (a) Mast markings
   (b) Embedded Hall effect sensors
   (c) Vertical position Sensors
   (d) Laser ranging

   was undertaken based on knowledge gained in automated factories, marine labs, government reports from DOE and NASA, and experience in precise position sensing for underground nuclear test monitoring. After risk reduction considerations were introduced to the selection process, an off the shelf laser ranging system using time of flight principles that had previously been integrated with a Proportional-Integral-Differential (PID) controller for boom management was selected.

This approach to trade studies and design knowledge transfer is well suited to projects with multi-discipline teams, tools for performing multi-attribute utility analysis and in need of a consistent basis for establishing design criteria and evaluation of alternate design choices.

5 Conclusion

Re-usability of design and knowledge from one environmental area to another is aided by use of object oriented technology approaches, scenario analysis, credible simulations, design knowledge libraries, and various classes of reuse tools. However caution is required since powerful tools require care and experience in their use on projects with mission and safety critical aspects. Emplacing the infrastructure to support this approach is best if supported up front by management.

1. The scenario begins with the rover assigned to a maintenance action.

2. The manipulator element mates with the rover base.

3. Under teleoperated control, the combined rover moves to a point providing access for the best video of a failed system or SRU.

4. A maintenance engineer controls the combined arm and video system during the inspection task to diagnose the failure and plan for most effective repair.

5. Once a repair plan is generated, the mobile base with its attached arm moves to the commanded location of the spares supply.

6. The arm is used to load the required replacement SRUs on the base according to access requirements for the planned maintenance actions.

7. The base with arm and payload navigates to the appropriate point to start system repair action.

8. The arm controlled by the teleoperator with inputs from video and supporting analysis to locate the failed module.

9. It removes the module and returns it to the equipment carrier on the base.

10. The arm locates the replacement SRU and places it in the operational configuration.

11. After all maintenance and inspection actions are completed, the base traverses to the failed module crib [or equipment airlock].

12. It removes the failed items from the carrier and places them in the appropriate location.

13. When demate of the manipulator arm and its systems from the mobile base is complete, the scenario ends.

Figure 1: Lunar Outpost Maintenance Scenario

Name:
Life Cycle Phase:
Type:
Purpose:
Maturity:
Risk Characterization:

- **Contexts:**
  - **With:**
  - **Note:**
  - **Rationale:**
- **Provided Interface:**
  - **Resources:**
    - **Name:**
    - **Constraints:**
    - **Exceptions:**
    - **Notes:**
    - **Rationale:**
  - **Services:**
    - **Name:**
    - **Constraints:**
    - **Exceptions:**
    - **Notes:**
    - **Rationale:**
- **Required Interfaces:**
  - **Resources:**
    - **Name:**
    - **Constraints:**
    - **Exceptions:**
    - **Notes:**
    - **Rationale:**
  - **Services:**
    - **Name:**
    - **Constraints:**
    - **Exceptions:**
    - **Notes:**
    - **Rationale:**
- **State Constraints:** ...

Figure 2: Structured Specification for Design Reuse Browsing

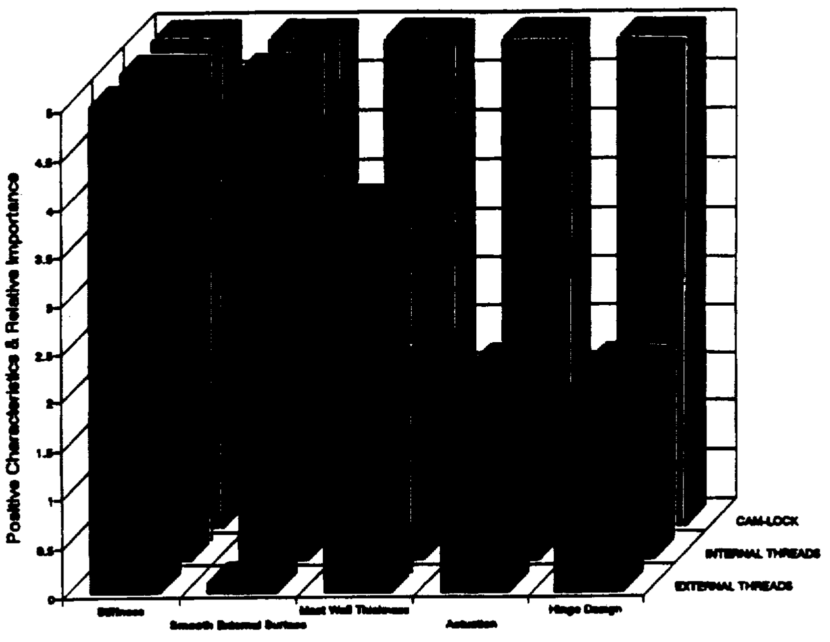


Figure 3: CAM LOCK Trades

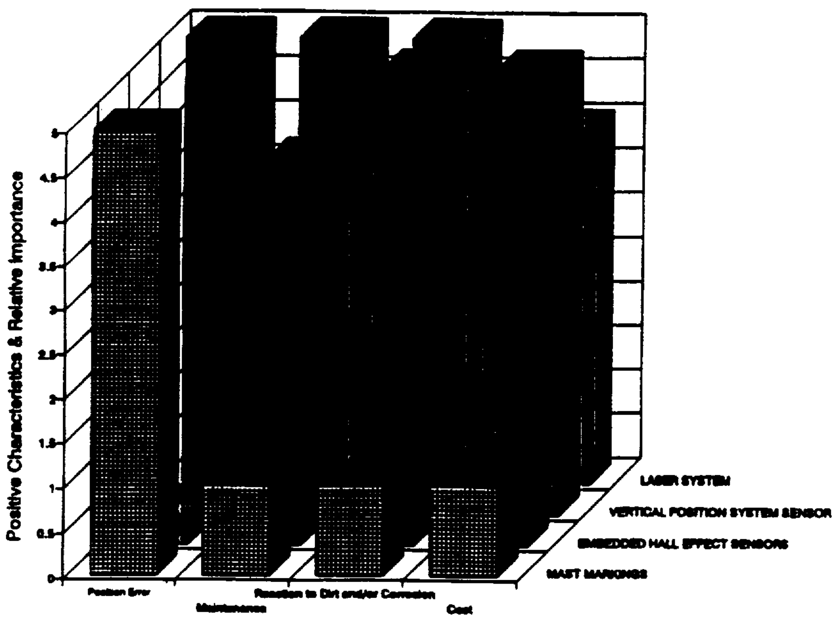


Figure 4: Mast Position Sensor Trades

AIAA-94-0888-CP

"A MULTI-MODE MANIPULATOR DISPLAY SYSTEM FOR CONTROLLING REMOTE ROBOTIC SYSTEMS"

N94-30569

Michael J. Massimino*
Michael F. Meschler**
Alberto A. Rodriguez**
McDonnell Douglas Aerospace
Houston, Texas

6064

20 1666

p. 7

Abstract

The objective and contribution of the research presented in this paper is to provide a Multi-Mode Manipulator Display System (MMDS) to assist a human operator with the control of remote manipulator systems. Such systems include space based manipulators such as the space shuttle remote manipulator system (SRMS) and future ground controlled teleoperated and telescience space systems. The MMDS contains a number of display modes and submodes which display position control cues position data in graphical formats, based primarily on manipulator position and joint angle data. Therefore the MMDS is not dependent on visual information for input and can assist the operator especially when visual feedback is inadequate. This paper provides descriptions of the new modes and experiment results to date.

1. Introduction

Manual control of a remote manipulator can be a difficult task due, in part, to a lack of useful feedback to the operator on the position of the manipulator with respect to its desired position, destination, or target object to be manipulated. For example, to control many remote manipulator systems, including the space shuttle remote manipulator system (SRMS), the operator relies largely on visual feedback from direct views through windows and indirect views from cameras. However, the visual information can be insufficient in providing the operator with adequate cues, due to obstructions, poor viewing angles, camera failures, or problems with resolution or camera control.

The Multi-Mode Manipulator Display System (MMDS) is being developed by MDA to alleviate some of these difficulties. The current design of the MMDS consists of two major modes: 1) the Manipulator Position Display (MPD) mode, and 2) the Joint Angle Display (JAD) mode. At the time of the writing of this paper, the MPD mode has undergone testing and is further along in the development cycle than the JAD mode which is in its initial development.

*Lead Research Engineer, Product Development
Member AIAA
**Senior Software Engineer, MDA

2. Manipulator Position Display (MPD) Mode

The Manipulator Position Display mode consists of two sub-modes: 1) Rotational/Translational (R/T) Submode, and 2) MPD Pilot Submode. The two submodes of the MPD were designed to help alleviate the problems associated with poor visual feedback caused by obstructions, poor viewing angles, poor resolution, camera control, or camera failure. This can be done because the MPD does not rely on visually obtained information as a source of input, but rather on six degree of freedom position information data from the manipulator system sensors (for example, joint position encoders).

Further, with the MPD displays, six degree of freedom position cues are displayed to the operator in a graphical format. The MPD displays the six degree of freedom cues concurrently. In addition, the MPD's algorithm performs the necessary calculations and provides the operator with "fly-from" or "fly-to" cues that alleviate the burden of calculating the appropriate system inputs from the operator. [1]

The MPD needs to know the current and desired (or target) positions. The current position of the manipulator arm can be obtained through real time position data from the manipulator arm in six degrees of freedom. The desired position of the arm in six degrees of freedom needs to be identified and entered into the MPD program. With this knowledge, the MPD displays can present the deviation or error that exists in each degree of freedom to the operator in an easy to use format. The MPD displays not only have applications for the SRMS, but also for other human-machine applications (aircraft, deep sea manipulators, nuclear environment, etc.) which require the operator to control multi-degree of freedom systems under limited viewing conditions when desired target points are known.

The experiments conducted with the two submodes of the MPD mode showed that using either submode significantly improved operator performance (by 25 to 33%) over performing the same manipulation tasks without the use of the MPD submodes.[2,3]

2.1 Rotational/Translational Submode

Figure 1. shows the format of the Rotational/Translational (R/T) Submode of the Manipulator Position Display mode.[3] The Rotational/Translational Submode separates the rotational and translational cues to be represented by the motion of two separate objects. This submode was designed so that one object on the display would correlate exclusively to the translational inputs on the hand controllers, while the second object would correlate exclusively to rotational inputs on the hand controllers.

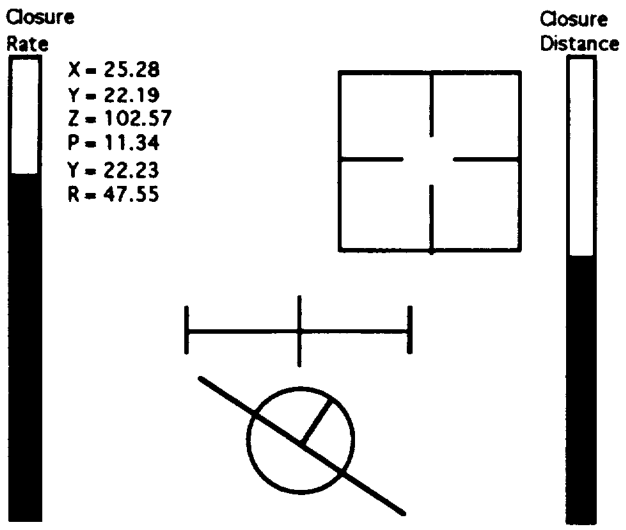


Fig. 1. MPD Rotation/Translation Submode Format

The line in the center with the three tick marks in Fig. 1 is stationary and acts as the reference line. The operator drives the translational cues using the square with the tick marks shown in Fig.1. Deviation in Z-translation is depicted by the square being above or below the reference line, while Y-translation deviation is shown by the square being to the left or right of the center of the reference line. For X-translation, the operator relies on the size of the square relative to the length of the reference line. For rotational cues the operator would look to the circular object shown in Fig. 1. The position of the circle with respect to the reference line provided the rotational deviation information to the operator. If the circle is above or below the reference line, a deviation in pitch exists. A deviation in yaw is depicted by the circle being to the left or right of the center of the reference line. Roll cues are provided by the orientation of the extended line running through the center of the circle and the shorter line in the center of the circle. If those lines are tilted to the left or to the right, then a deviation in roll exists.

2.2 MPD Pilot Submode

The format of the MPD Pilot Submode of the Manipulator Position Display is shown in Fig. 2.[4] The MPD Pilot Submode got its name because it utilizes cues, such as a yaw ball and a pitch horizon line, similar to those found in aircraft. The line in the center with the three tick marks is stationary and acts as the reference line. The operator drives five of the six position/orientation cues to that reference line, all except the yaw cue which is shown separately at the bottom of the display.

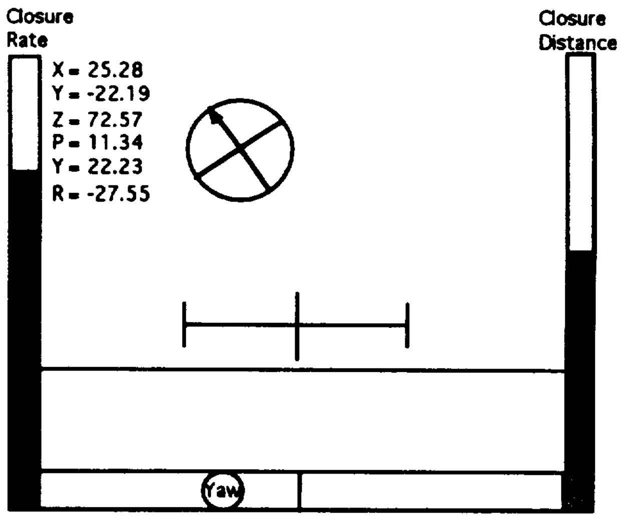


Fig. 2. MPD Pilot Submode Format

All deviations in the translational degrees of freedom are displayed by the circle with the crosshairs inside of it. If the crosshairs are to the left or right of the center of the reference line, a deviation in Y-translation exists. A deviation in Z-translation is depicted with the circle and crosshairs being either above or below the reference line. Errors in X-translation is depicted as a size difference between the circle with crosshairs and the length of the reference line. For rotational cues the operator would look to the yaw ball at the bottom of the display, the horizontal pitch line (shown just below the reference line in Fig. 2), and the orientation of the crosshairs in the ball for roll information. The error in the yaw degree of freedom are shown by the yaw ball in Fig. 2 being to the left or right of center. Pitch error is shown by the horizontal pitch line being above or below the reference line. For roll cues the operator uses the orientation of the crosshairs inside the ball.

In addition, for both of the submodes discussed, the operator is provided with a digital readout of the deviations in each of the six degrees of freedom. This digital readout can be seen in the upper left hand corner of Figs. 1 and 2, and is helpful in the final stages of a task to ensure that the deviations are within the desired limits (i.e. close to zero).

Both submodes also contain two bar graphs on either side. The bar graph shown on the left of Figs. 1 and 2 provides rate information, and the bar graph on the right of Figs. 1 and 2 provides the absolute closure distance between the current manipulator position and the desired manipulator position. This information can be particularly helpful to control the rate of movement based on the distance from the target location. For example, if the manipulator were far from the target location the operator would probably want to moving faster than if the manipulator was very close to the target location.[5,6]

2.3 Experimental Results with the MPD Mode

To quantify the effectiveness of the two submodes of the MPD described in the previous sections, experiments with human operators were conducted. The MPD display submodes were presented to four trained and experienced test subjects on a GRID 1660 laptop computer. A space shuttle SRMS task was simulated using the Manipulator Analysis - Graphic, Interactive, Kinematic (MAGIK) simulation system which runs on Silicon Graphics computers. The task was a space station assembly task, which focused on the installation of a Pressurized Mating Adapter (PMA) to a space station module.

Three experimental conditions were tested: 1) performing the task with the aid of the Rotational/Translational Submode of the Manipulator Position Display, 2) performing the task with the aid of the MPD Pilot Submode of the Manipulator Position Display, and 3) performing the task without the aid of the MPD display mode. For all three experimental conditions the operators were given the clearest available camera view of the task (simulated by the MAGIK system)[7]. In addition, the operators were also given a digital readout of the position of the manipulator in each degree of freedom through a simulation of the SRMS display panel. During the experimental condition of performing the task without an MPD display, this digital position information was critical for the final steps of the task when the camera view became less helpful.

Each test subject completed training for performing the task without the MPD and with each submode of the MPD. Training ended when the test subject's performance times reached steady values and learning curves flattened. Three separate experimental sessions were conducted for each subject. During one experimental session, the subject performed the task without the MPD display , in another session the subject performed the task with the MPD Pilot Submode, and in a third session the subject performed the task with the Rotational/Translational Submode. At the start of each experimental session, each subject was given warm-up trials and then six to ten data trials were conducted. The subject could end an experimental trial when the deviation in each translational degree of freedom was less than 1 inch, and the deviation in each rotational degree of freedom was less than 0.5 degrees.

The mean task times for performing the tasks under the three experimental conditions are shown in table 1.

Pilot Submode	No MPD	R/T Submode
3.9 min	5.2 min	3.5 min

Table 1. Mean task times.

Fig. 3 shows the total average task times calculated across all of the four test subjects. The total mean task time averaged for all four test subjects was 3.9 minutes with a mean standard error of 0.12 minutes when using the MPD Pilot Submode , 5.17 minutes with a mean standard error of 0.19 minutes when not using the MPD, and 3.54 minutes with a mean standard error of 0.14 minutes when using the Rotational/Translational Submode. Therefore, the Rotational/Translational Submode provided an average improvement of approximately 33% while the MPD Pilot Submode provided an average improvement of approximately 25%. These results were statistically significant to the 99% confidence level.

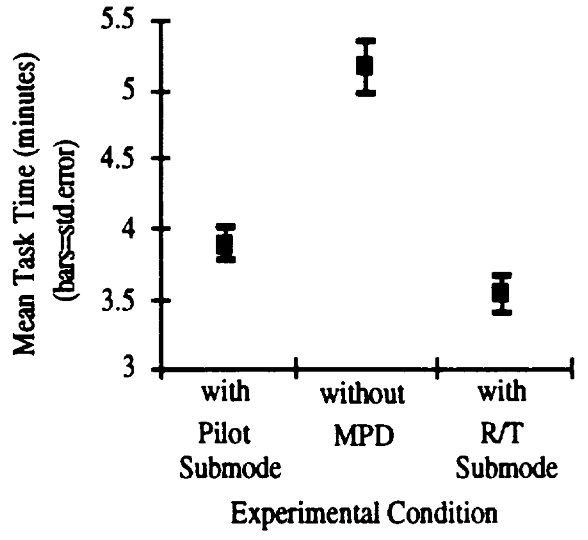


Fig. 3. Average of experimental results for all test subjects.

3

A statistical analysis with a series of paired t-tests showed that using the MPD Pilot Submode significantly improved operator performance at the 99% confidence level (t(30)=7.44, p<0.01). A series of t-tests were also conducted to determine the statistical significance of using the Rotational/Translational Submode versus not using the MPD. As was found with the MPD Pilot Submode, the Rotational/Translational Submode significantly improved operator performance at the 99% confidence level (t(30)=7.41, p<0.01). The statistical analysis of the results of using the Pilot Submode versus using the Rotational/Translational Submode produced differing conclusions based upon individual test subject performance. Test subjects #1 and #3 performed significantly better with the Rotational/Translational Submode than with the Pilot Submode. However, for test subjects #2 and #4, there was no significant performance difference between using the Rotational/Translational Submode versus using the Pilot Submode. The total average over all 4 test subjects did show a significant performance advantage with the Rotational/Translational Submode over the MPD Pilot Submode (t30)=2.06. p<0.05).

2.4 Advanced Features of the MPD

As a result of the experiments described above and comments from astronauts, mission designers, and astronaut trainers, a number of recommendations for improving the MPD were gathered. These recommendations have resulted in the implementation of a number of new features. The following section describes each of the new features and their benefits.

2.4.1 Highlighting

One advanced feature is highlighting cues to help the operator distinguish between the lines which represent the rotational and translational cues, and the stationary reference line at the center of the screen. This feature becomes most useful when the manipulator is reaching its target position and the operator is trying to align the cues to the stationary reference line. This is one of the most critical phases of any operation.

For each task there are defined tolerance limits, for each degree of freedom, within which the manipulator is considered to be at its desired final position. Based on this information a highlighting feature was implemented which indicates to the operator when the manipulator is within the defined limit for each degree of freedom. This indication is achieved by increasing the width of specific lines on the rotational and translational cues when the manipulator position and attitude are within the specified range. For example, when the Point of Resolution (POR) of the manipulator is within the specified range in the X-axis (see figure 6-8) the square, in the R/T Submode, will become bolder than the other lines. In turn, when the POR of the manipulator is within tolerance in the Y-axis the vertical lines in the translational cue will become bolder. And finally, when the POR is within the limit in the Z-axis, the horizontal lines of the translational cue become bold. Once all of the lines which comprise the translational cue are bold, the operator will know that the manipulator tip is within tolerance in the X, Y, and Z axes.

For the rotational cues in the R/T Submode, the circle becomes bold when the manipulator's POR is within the yaw limit. The horizontal line drawn through the circle is made bold when the pitch limit is satisfied. And the vertical roll indicator is made bold when the POR is within the roll limit. As with the translational cue, when the manipulator POR is within limit in yaw, pitch, and roll the entire rotational cue will be bold. Fig. 4 shows an example of the bold feature indicating that the X-axis and the yaw axes are within range. The tolerances can be set to different values for each degree of freedom. This feature is also implemented in the MPD Pilot Submode.

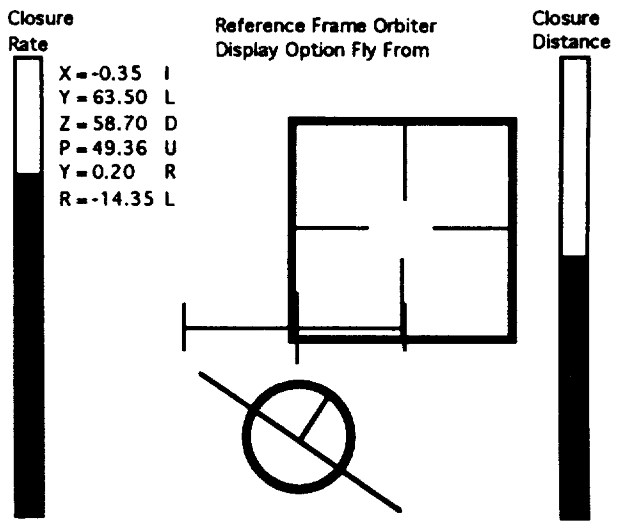


Fig. 4. MPD highlighting feature.

Another benefit of using line width as an indication of reaching final position is the ability to reach the desired position in any one axis regardless of where the cues are on the screen with respect to the reference line. For instance, in the event that a particular translational axis needs to be aligned before the other axes this can be done without the translational cue being lined up with the stationary reference line. This occurs when a payload must be centered in the X and Y axes before being lowered into the shuttle bay. During such a task the operator would have to adjust the size of the square to coincide with the length of the reference line without having the translational cue over the reference line. Without the added feature this would be accomplished by referring to the deltas being displayed on the upper left-hand corner, recalling the defined limit for each axis, and watching the translational cue. With the added feature the operator need only concentrate on the translational cue receiving a visual signal when the POR is within range for the desired axis(in this example the X-axis).

2.4.2 Color Cues

In addition to the highlighting feature, the MPD display now provides color cues to help distinguish between the translational and rotational cues, and the stationary reference line. The use of color is useful when the manipulator POR is close to its final destination as shown in Fig 5. As can be seen in the figure it can be difficult to differentiate between the translational cue, rotational cue, and the reference line. In the current MPD implementation the translational cue is drawn in red, the rotational in green and the reference line in white.

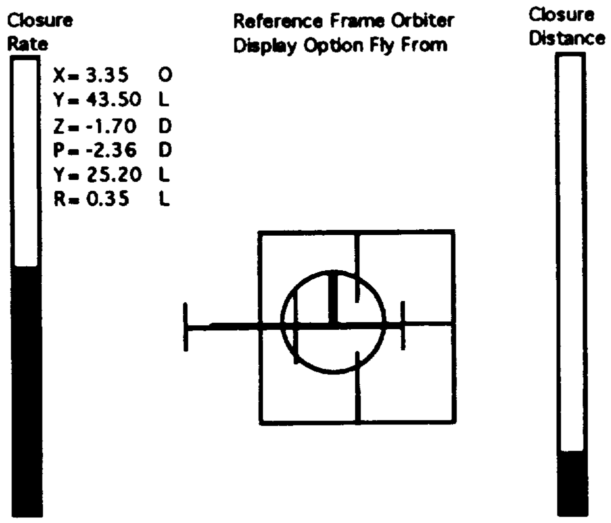


Fig. 5. MPD Display need for color cues.

Color cues are also being considered in conjunction with the highlighting feature to give the operator information on the proximity to the final destination. The idea is to define a range, like the limits described in the previous section, which when entered by the manipulator POR would cause the translational and rotational cues to change color. This would supply the operator with a visual cue that the manipulator POR is reaching its destination and in turn the hand controller inputs should be reduced in order avoid going beyond the desired final position. Once the previously described final limits are reached, the translational and rotational cues' colors can again be changed as the lines get bold. In this way the operator is given two signals that the manipulator has reached the final POR, bolder lines and change in color. [9]

2.4.3 Direction Cues

In the original implementation of the MPD, the deltas between the current and final POR positions were displayed as signed numbers in the upper left-hand corner of the screen. The sign of the numbers is provided as an indication of the direction in which the delta exists. In Fig. 5 this can be seen in the Z-axis and pitch digital delta readouts. This approach required the operator to mentally transform the sign cue to the coordinate frame in which they are working, then figure out the corresponding hand controller deflections required to compensate for the deviation. However, what usually occurs is that the operator inputs the wrong direction based on the sign delta.

To alleviate this difficulty the MPD includes a feature referred to as "Direction Cues". Direction Cues supply the operator with instructions of the necessary hand controller deflections to remove the deltas in each degree of freedom. The Direction Cues can be seen in Fig. 5 as letters following the deltas in the upper left-hand corner of the display. The letters I or O are used to indicate in or out deflection of the translational hand controller, L or R for left or right deflection of the translational hand controller, and U or D for up or down deflection of the translational hand controller. For the rotational Direction Cues the letters U, D, L , and R are used in the same way as with the translational Direction Cues. Fig. 5 shows the display signaling the operator to deflect the translational hand controller out, left, and down and the rotational hand controller down, left, and left for the pitch, yaw, and roll axes respectively. With the addition of Direction Cues the operator is presented with straight forward indications of the necessary hand controller deflections eliminating the possibility of unnecessary and potentially dangerous movement of the manipulator.

2.4. 4 Fly-To/Fly-From Option

The original version of the MPD displays used what is referred to as "fly- from," or outside-in, convention to show the deviation between the current manipulator POR position and the desired final position. In the fly-from convention the objective is to input the necessary hand controller deflections to move the graphical cues from their current positions to a specified reference point in the display. In the MPD displays the reference point is the stationary reference line in the center of the screen. As operators with varying backgrounds used the MPD displays two points were made about the utilization of the fly-from convention.

First, it was not obvious from the information presented by the MPD displays that a fly-from convention was being used. And secondly, not everyone is used to the fly-from convention. Some operators are more comfortable with the "fly-to", or inside-out, convention. In the fly-to convention the objective is to deflect the hand controllers in such a way as to move a specified reference point, the stationary reference line, to the current position of the graphical cues. As the hand controller inputs are generated the graphical cues move towards the reference line giving the illusion that the reference line is moving. [10]

Having reached the conclusion that neither one of the conventions exhibit any inherent advantages, the MPD display now gives the operator a choice of using either option. At the beginning of each task the operator selects whether the graphical cues are shown in the fly-to or fly-from convention. Once this selection is made, the MPD displays the option in the top center part of the screen as can be seen in Fig. 5. This new feature gives the flexibility to use the display in the convention which is most comfortable to the operator and also makes the current selection obvious at all times.

2.4.5 Coordinate Frame Selection

The last addition to the original MPD display is the capability to select between the different coordinate frames in which to command the manipulator POR. Originally the commands where all based in the orbiter coordinate frame which is shown in Fig. 6.

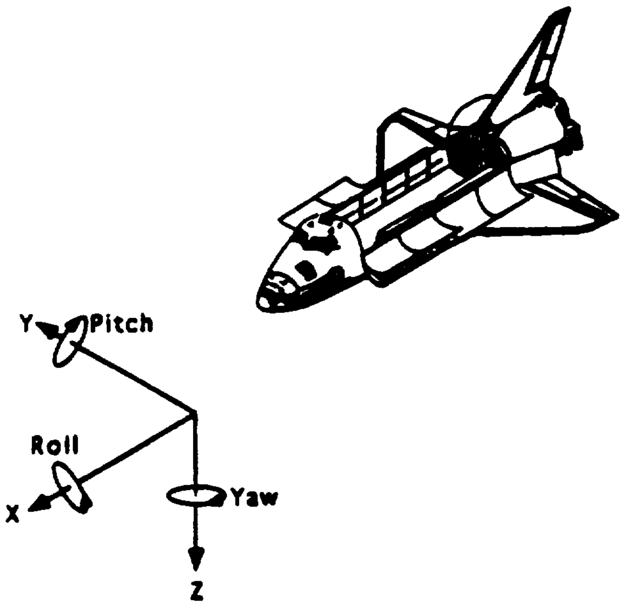


Fig 6. Space Shuttle coordinate reference frame.

With the addition of the coordinate frame selection feature the operator now has a choice between orbiter, end effector, and payload coordinate frames. In the case of the space shuttle, this is a major improvement over the information currently displayed in the aft flight deck which is always in orbiter reference mode. An example of an end effector coordinate frame is depicted in Fig. 7.

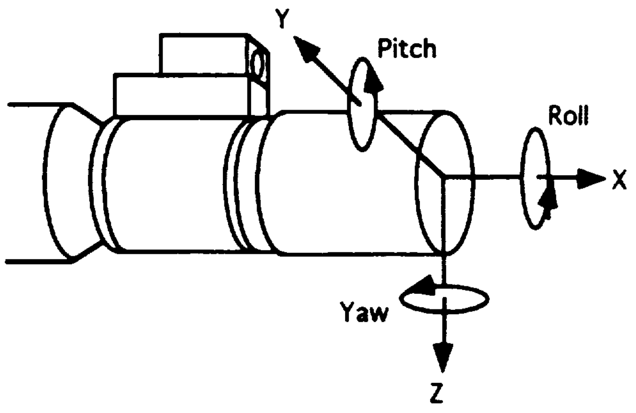


Fig. 7. End effector coordinate frame.

The payload coordinate frame is different for each payload and can sometimes coincide with either the end effector or orbiter coordinate frames. Fig. 8 shows an example of a payload coordinate frame.

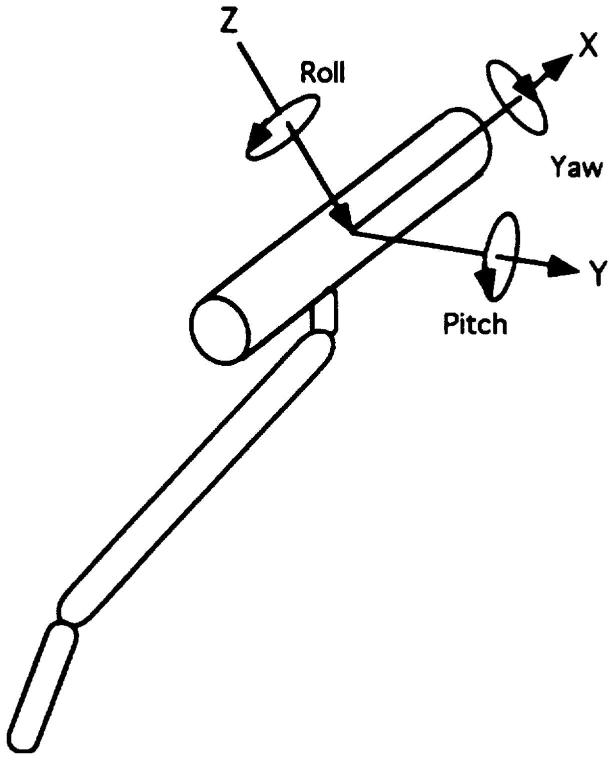


Fig. 8. Payload coordinate frame.

The coordinate frame selection feature provides consistency in the way the graphical cues display changes in the different axes. For example, in the R/T Submode movement in the X-axis is always depicted as changes in the size of the square of the translational cue. Motion in the Y-axis is always shown as a change in the translational cue's horizontal position on the screen. And motion in the Z-axis is always shown as a change in the translational cue's vertical position on the screen. The selected reference frame is displayed in the top center part of the screen (see Fig. 5). [11]

3. Joint Angle Display Mode

The second major mode of the MMDS is the Joint Angle Display (JAD) Mode. The JAD is comprised of a set of bargraphs which represent the position of each joint of a manipulator. The JAD mode has three submodes: 1) nominal operations, 2) joint limits, and 3) single joint operations.

3.1 Nominal Operations Submode

The nominal operations mode displays the current joint positions to the operator. For example, the SRMS has six joints as is shown in Fig. 9. Fig. 10 shows how the six joint values for the SRMS would be presented to the operator. Note that each joint in Fig. 9 is listed in Fig. 10. Each bar graph represents the current joint angle. The bar graphs are updated in real-time based on the changing encoder values at each joint.

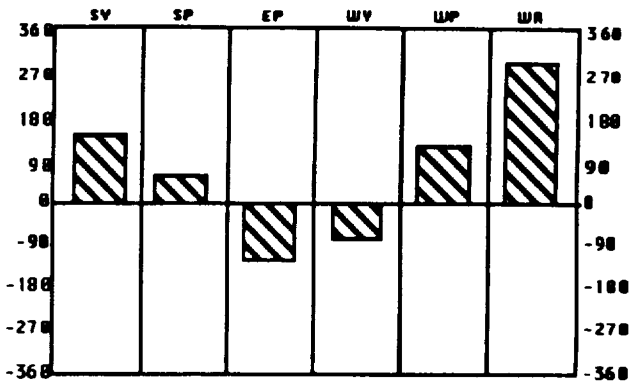


Fig. 10. Nominal operations Joint Angle Display

3.2 Joint Limits Submode

The second submode of the display will include all the features of the first submode plus cues to indicate the location of the joint limitations. As can be seen in Fig. 11 the joint limits are depicted by the small triangles to the right of each bar graph. For instance Fig. 11 shows that for the SY joint the joint limits are at ±180°. This display could also emit an audible tone when any joint reaches a limit. By including the audible tone the operator will be notified of a joint limit error without having to constantly monitor each joint. Having the features designed in this submode of the JAD provides the operator with a tool to avoid joint limits.

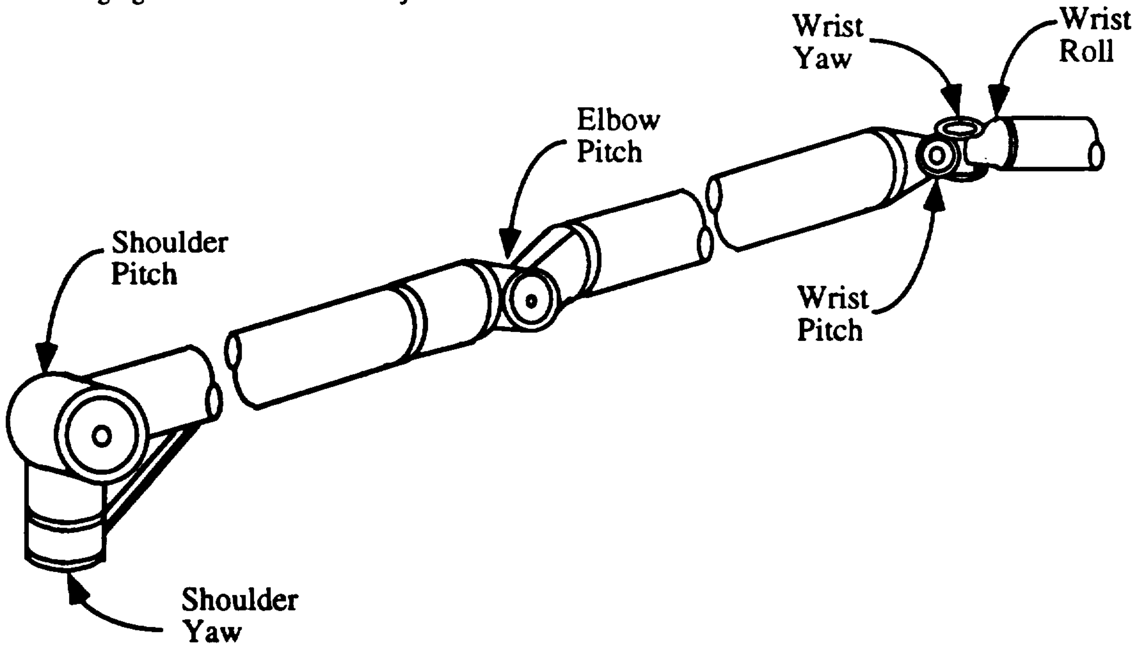


Fig. 9 SRMS Manipulator

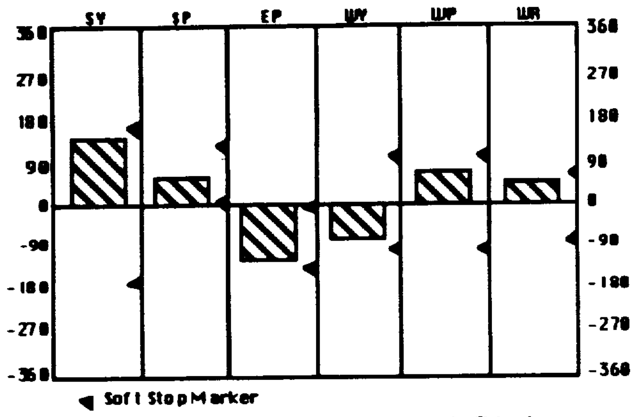


Fig. 11. Joint Limits in the Joint Angle Display

3.3 Single Joint Operations Submode

Another application for the JAD will be single joint operations when the operator needs to drive the arm one joint at a time. This operational scenario occurs on the space shuttle during failure modes which make controlling all joints concurrently impossible (for example, a hand controller failure). During these operations, the Single Joint Operations submode will not only provide the operator with information on the current joint positions and joint limits, but will also provide the operator with operational cues. These cues will include the amount of deflection needed for each joint, and the joint sequence. One limitation of this display is, however, that the encoder data from the manipulator joints are needed to run the display and might not be available in the event of a failure.

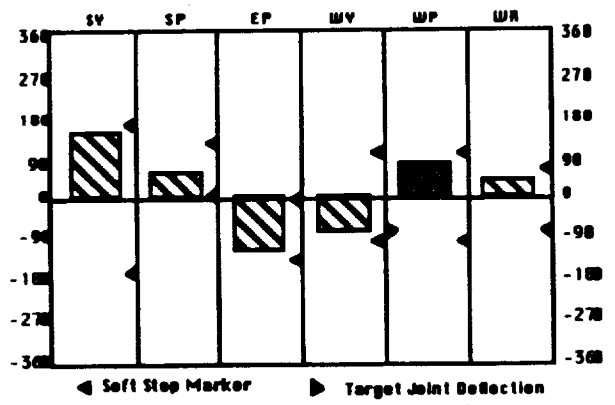


Fig. 12. Wrist pitch joint indication.

Fig. 12 provides an example of the Single Joint Operations Submode display. Fig. 12 indicates that the Wrist Pitch joint should be moved to -86 degrees. Once the operation in Fig. 12 is complete, the next step would be displayed.

4. System Summary

With the completely integrated MMDS the operator is supplied with a complete, concise, and flexible view of the state of the manipulator at all times. This complete view includes information on both the manipulator POR position through the use of the MPD displays, and the position of each individual joint through the use of the JAD. Using the MMDS, a typical grapple and unberth task with SRMS can be described as follows.

The operator begins the task using the MPD display of their choice, Pilot or Rotational/Translational Submode, in end effector coordinate reference frame and fly-from mode. As the operator maneuvers toward the grapple fixture, they can at any time switch to the JAD viewing the status of each joint and their proximity to any limits. Once the POR is within the predefined limits the translational and rotational cues are highlighted. At this time the payload is grappled and the operator switches to orbiter coordinate reference frame.

With the payload grappled a new target POR position is entered and the translational and rotational cues adjust to show the new deltas. The operator now begins to issue the appropriate hand controller deflections to move the manipulator towards the new destination. If at any point during the task a joint limit is reached, the JAD will sound an audible tone anunciating that such a limit has been reached.

Upon recognizing the joint limit alarm, the operator will switch to the JAD where he/she can rapidly identify the errant joint. The operator would then switch to single joint mode and command the wayward joint away from the limit using the JAD. Once the joint is backed away from the limit the operator can revert to the MPD display to reach the final POR.

Another task would be to berth the payload into the orbiter bay. Once again the new target position is entered and the translational and rotational cues adjusted to show the deltas. At this point the operator can use the payload coordinate reference frame to drive the payload into its berthed position. Once the final berthed position is reached the task is completed.

One final note with respect to the flexibility of the MMDS. At any time during the task described above the operator has the capability to choose between how and what information is displayed without having to restart the MMDS. The operator can switch between the MPD or JAD, Pilot or R/T Submode, and coordinate reference frames. This capability gives the operator the ability to command the manipulator in a way that is most suitable to their background and training.

5. Conclusions

Based on the development and experimental results presented in this paper, the MMDS can be expected to provide significant operational benefits including providing the operator with useful manipulator position information, reducing control problems associated with the poor viewing conditions, reducing operator workload, reducing training time, and assisting the operator with performing unscheduled or unpracticed procedures. The MMDS has space based application for the SSRMS space station as well as for ground control of space based manipulators. Its potential application areas will hopefully be expanded into environmental, hazardous waste, nuclear, and undersea remote manipulation environments.

6. References

1. D.W. Collins, "Payload Deployment and Retrieval System Overview Workbook," NASA NASA TD383, NAS9-18000, NASA Johnson Space Center, Mission Operations Directorate, Space Flight Training Division, Houston, TX, Feb. 1988.

2. M. Massimino and M. Meschler, "Experimental Results Using a Manipulator Position Display as a Human-Machine Interface Aid for Controlling Space Robotic Tasks," AIAA Space Programs and Technologies Conference and Exhibit, Huntsville, AL, Sept. 1993, paper # AIAA 93-4114.

3. M.J. Massimino, M.F. Meschler, and A.A. Rodriguez, "Human-Machine Interface Aids for Space Telerobotics," 44th Congress of the International Astronautical Federation, Graz, Austria, Oct. 1993, paper # IAF/IAA-93-G.3.154.

4. M.J. Massimino, et. al., "Manipulator Position Display (MPD) for Human-Machine Interaction," Telemanipulator Technology and Space Telerobotics, SPIE Proceedings Vol. 2057, SPIE's International Symposium on Optical Tools for Manufacturing and Advanced Automation, 7-10 September 1993, Boston, MA.

5. E.C. Poulton, Tracking Skill and Manual Control, New York: Academic Press, 1974

6. M.J. Massimino, T.B. Sheridan, and J.B. Roseborough, "One Handed Tracking in Six Degrees of Freedom," 1989 IEEE International Conference on Systems, Man, and Cybernetics, Conference Proceedings, Vol. 2, pp. 498-503. Cambridge, MA, 14-17 November, 1989.

7. R.G. Boettger, K.E. Harvey, A.S. Mediavilla, W.C. O'Donnell, "Manipulator Analysis - Graphic, Interactive, Kinematic (MAGIK) Version 5.4 User's Guide", NASA TM-5.24.11-25, NASA Johnson Space Center, Automation and Robotics Division, Houston, TX, Sept. 1992.

8. "Space Station Freedom Manipulator Assembly Sequence Assessment," NASA doc. no. JSC-25554 Rev.A, NASA Johnson Space Center, Automation and Robotics Division, Houston, TX, July 1993.

9. E. L. Wiener, D. C. Nagel, Human Factors In Aviation, New York, Academic Press, 1988.

10. L. R. Young, "Human Control Capabilities," Bioastronautics Data Book, Second Edition, pp 751-806, NASA Scientific and Technical Informarion Office, Washington, D. C., 1973.

11. J. J. Craig, Introduction to Robotics Mechanics and Control, Second Edition, New York, Addison-Wesley Publishing Co., 1989.

12. "PDRS Operations Checklist," NASA doc. no. JSC-48039, NASA Johnson Space Center, Mission Opreations Directorate Systems Division, Houston, Tx. July 30, 1991.

Acknowledgments

The authors thank the McDonnell Douglas Independent Research and Development Program under the direction of Mr. John Hill for funding this research, and Brian Richard, Pat Bevill, Ken Ruta, and John Dunn for participating as test subjects.

S14-63

6065

AIAA-94-1221-CP

207337

p 6

PROGRAMMABLE AUTOMATED WELDING SYSTEM (PAWS)

N94-30570

Martin D. Kline
Project Manager
Babcock & Wilcox, CIM Systems
Lynchburg, VA

Abstract

An ambitious project to develop an advanced, automated welding system is being funded as part of the Navy Joining Center with Babcock & Wilcox as the prime integrator. This program, the Programmable Automated Welding System (PAWS), involves the integration of both planning and real-time control activities. Planning functions include the development of a graphical decision support system within a standard, portable environment. Real-time control functions include the development of a modular, intelligent, real-time control system and the integration of a number of welding process sensors.

This paper presents each of these components of the PAWS and discusses how they can be utilized to automate the welding operation.

Introduction

The demands of small batch operations and the need to integrate into a wider automation strategy have pushed the development of advanced robotic and process control systems. One such system, presently directed specifically at welding applications, is under development by Babcock & Wilcox. This approach addresses integrating both off-line planning and real-time control activities. This system was initially developed as an Advanced Technology Development program with the Naval Surface Warfare Center, Carderock Division, and is presently in an industrial transition phase as part of a Navy ManTech contract. This ManTech program is coordinated through the Navy Joining Center in Columbus, Ohio.

This advanced control system, known as the Programmable Automated Welding System (PAWS), has been created specifically to provide an automated means of planning, controlling, and evaluating critical welding situations to improve productivity and quality. The primary issue has been the desire to increase the cost-effectiveness and applicability of automation to difficult welding situations.

System Overview

PAWS consists of an Off-line Programming System (OLP) and an on-line, real-time controller. The OLP system provides a means to develop the plan for an entire automated welding operation, as well as the capability to manage existing plans. The OLP system provides an integrated platform for the motion and process planning functions. The Controller is capable of then implementing these plans during the actual welding process.

PAWS Off-line Programming System

The PAWS-OLPS resides on a UNIX-based workstation and is comprised of a relational database, a motion planning module, a geometric modeling system, and a job builder module. This system was developed following a client-server philosophy specifically to provide a decision support tool for the development, storage, and management of programs for the PAWS controller. The use of standards and the requirements of hardware portability have been highly stressed.

The PAWS-OLPS provides a means to develop and plan an entire automated robotic welding operation based on a computer aided design (CAD) model of the part to be welded. This planning occurs away from the robot and allows the robot system to maintain production while new applications are being planned and verified.

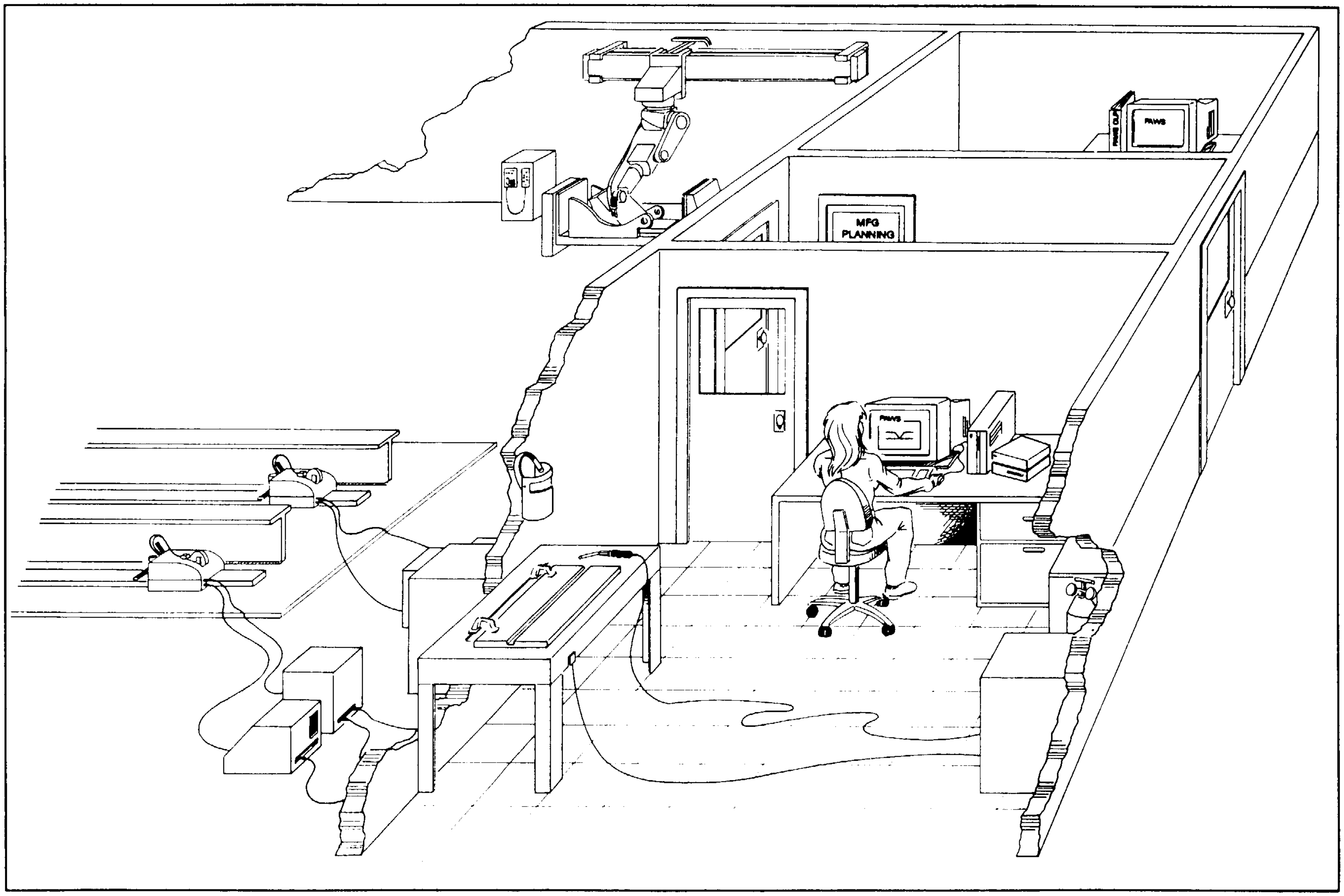


Figure 1: Programmable Automated Welding System

One OLPS can support multiple cells. OLPS is a key technology in small batch robotic operations. Without an OLPS, the economical utilization of robotic systems in small batch operations is not possible due to the non-productive time required for the iterative nature of manual teaching methods.

The PAWS-OLPS provides a highly integrated approach to the planning of the welding and sensor operations. The welding application data is maintained in a relational database that is tightly coupled to the operation of the OLPS and the development of the overall plan. Storage of certified welding procedures in a standard database format allows for the maintenance and re-use of previously performed welding trails. The client-server architecture allows the welding information to be maintained at a distributed computer by a knowledgeable resource. During subsequent planning operations, the OLPS presents to the planner only welding selections which are appropriate for the application at hand. By encapsulating and abstracting the welding knowledge, the PAWS-OLPS practically eliminates the need for the OLPS planner to be knowledgeable of the details of the welding process.

The OLPS also incorporates advanced motion planning and optimization methods. Functions are provided for optimal placement of the robot with respect to the workpiece and for automated determination of a collision-free path. Both the placement optimization and the collision avoidance capabilities are technologies critical to supporting small batch operations where accessibility limitations exist. The motion simulation ties the motion of the manipulator to the process information. This module provides a graphical 3-D animation of the manipulator performing the welding operation with real-time collision detection. A geometric modeler provides a convenient method for the modeling of parts, manipulators, end effectors, and physical environment constraints. The OLPS also provides a means of importing CAD files of components and generating solid models from those files. Once the plan is determined to be satisfactory, the plan is converted into a job to provide true off-line programming of the entire welding operation. This job is provided to the PAWS controller in the form of text files which are then converted to the controller's real-time database format.

Figure 1 depicts a setup in which the OLPS is being utilized to plan the operations for multiple devices. This concept envisions two track devices performing simple linear welds and an inverted robot arm coordinated with the motion of a 2-axis positioner. The welding engineer is developing and documenting procedures on a computer based in the weld lab. This interface enables the sharing of historical information and provides a dynamic means of managing weld procedures. Motion planning and simulation is then performed by the manufacturing planning department. The welding knowledge is referenced during this process.

PAWS Controller

The PAWS real-time robotic controller represents a state-of-the-art system based on a VME multi-processor platform. On-bus resources provide the interface to the process equipment via industrial I/O (digital and analog), system I/O (serial and network interfaces, hard disk, monitor, etc..), and servo motion boards. This environmentally hardened platform supports the control of any common robotic manipulator and any arc welding process. In addition, the system has been designed to simultaneously accept input from a host of real-time sensors.

The key element of the controller is its flexibility. The controller and supporting peripheral components can be scaled to the technical needs of the application. This feature is supported both in software and by the ability to add processing power as dictated by the application. For example, the controller can be employed to control a simple 3-axis track mechanism or an articulated arm robot coordinated with a positioner. In addition, the controller can be configured to control a number of arc welding processes and sensors. In fact, given the proper welding equipment, multiple processes can even be maintained by a single controller. This flexibility enables the system to be extensible to emerging technologies such as new manipulators, welding processes and sensors.

The PAWS controller uses a real time database structure to compartmentalize the process data. The controller is comprised of both kernel and expansion modules (refer to Figure 2). The kernel modules provide the base technologies for robotic process control. The expansion modules can then be selectively employed to address the specific process needs (e.g. welding).

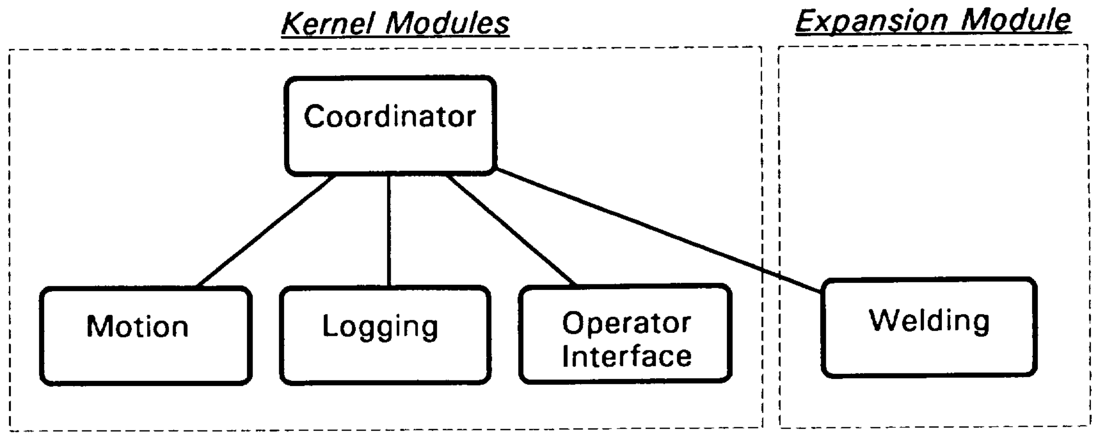


Figure 2: PAWS Controller Modules

Kernel Modules

Coordinator The coordinator module utilizes a script language to indicate the sequence of operation, and a rule-based expert system for exception handling. The sequence is built-up (either manually on the controller or automatically by the OLP's job builder module) as a series of statements specific to the process. These statements are English-like commands specific to the process at hand (e.g. START WELD, STOP WELD, MOVE ALONG, LOG DATA, etc..). This provides a readable, high-level view of the job plan. During execution, the exception handler monitors the state of the on-going process and issues programmed responses when anomalous conditions occur. These responses can range from simple warnings to complex adaptive responses.

Operator Interface All setup, walk-through teaching, and execution interaction with the operator is performed through an industrial pendant. This pendant consists of a portable, hand-held device with both push-buttons and a high-resolution graphical display. In appropriate situations, the pendant provides the operator the capability to adjust the process parameters during execution. To supplement the OLPS, an on-line editing capability is provided. The editor employs a portable PC and provides the ability to both edit and create jobs. In addition, this PC can be used to chart and analyze all logged data.

Motion The PAWS controller is capable of controlling a variety of manipulators, from a simple track device up to multi-axis robotic manipulators. A total of three manipulators and 32-axis may be controlled from a single controller. The 4 year ManTech program intends to develop and implement a production system in which a single controller is coordinating the operations of two or three robots simultaneously. Successful demonstration of this capability will provide a substantial cost advantage by sharing both hardware and manpower. A single operator can then be leveraged to monitor several operations at once.

The motion module also incorporates the ability to perform path memorization: to retrace, with or without an offset bias, a modified path. Additional features include: seam tracking, the ability to accept operator overrides of both Tool Y (cross seam) and Tool Z (standoff) distances, and the ability to modify motion parameters for adaptive control.

Logging The logging module allows for the selective logging of data based upon time, distance, or the reaching of an established threshold (e.g. heat input). Numeric data can be also be averaged while being logged.

Expansion Module

Welding The welding module commands the power supply to control the weld process. Currently, the system has been established to control the gas metal arc process (GMAW). The module commands and monitors a number of process parameters (e.g. current, voltage, wire feed speed, etc.). Process parameters are prevented from exceeding the limits established in the weld procedure. Additional general features include, consumable tracking and monitoring, user-definable I/O, and the expandability to other processes.

Adaptive parameter modifications are determined based upon input from sensors and from the operator. These may be direct parameter offsets (e.g. lower current 5 amps) or they may be in the form of indirect adjustments (e.g. increase bead width by 10%). Indirect adjustments are processed by the internal process model into the appropriate parameter offsets. This model also resolves conflicts between adjustment sources. This resolution is performed by evaluating both priorities and constraints. One elegant feature of this implementation is the ability for the operator to adjust the indirect parameters without knowledge of the necessary direct parameters. In other words, the operator can concentrate on physical characteristics, such as bead width, without needing to mentally calculate the necessary adjustments to current, voltage, and travel speed.

This module also handles all sensor interface issues. These include device-specific communications, user-programmable data filtering, and adaptive control of the motion module. Exception handling is performed by an embedded rules engine. The sensors currently being employed are listed in Table 1.

Table 1: Sensors

SENSOR	USAGE
Joint Vision Sensor	Seam Tracking Joint Volume Joint Shape
Post Weld Geometry Sensor	Pool Size Pool Location
Arc Element Sensor	Contamination in Arc (H_2, O_2, Fe)
Through-the-Arc Sensing	Seam Tracking
Touch Sensing	Joint Location

The listed sensors cover a wide range of control areas including feed forward, feedback, and process monitoring. The PAWS controller is capable, however, of being configured to utilize only those sensors which are needed to perform the particular application. A typical application which is severely space-limited may use only through-the-arc sensing, whereas, an accessible component with critical process control criteria may utilize three or four different sensors.

Summary

The 30 month-long ATD phase of the PAWS program ended in November 1992 and the follow-on 4 year ManTech program was started in September 1993. The system will be industrially hardened during the first year of this program and will be applied in an Navy Joining Center Teaching Factory at B&W CIM Systems in Lynchburg, VA. The technology will be implemented into
production systems during 1995. Follow-on years will focus upon expansion of the technology based upon end-user needs. This will include expansion into other welding processes (e.g. FCAW, GTAW, PAW), the support of multiple robots, expanded exception handling techniques, and the integration of design data directly into the OLP. In addition, the architecture is being developed for application to other non-welding robotic processes (e.g. inspection, surface finishing, cleaning).

6066

AIAA-94-1222-CP

ROBOTIC NDE INSPECTION OF ADVANCED SOLID ROCKET MOTOR CASINGS

S45-63
[illegible]
p. 13

N94-30571

Glenn E. McNeelege, Mechanical Systems Engineer
and
Chris Sarantos, Electrical Systems Engineer
Babcock & Wilcox, CIM Systems
Lynchburg, Virginia

Abstract

The Advanced Solid Rocket Motor program determined the need to inspect ASRM forgings and segments for potentially catastrophic defects. To minimize costs, an automated eddy current inspection system was designed and manufactured for inspection of ASRM forgings in the initial phases of production. This system utilizes custom manipulators and motion control algorithms and integrated six channel eddy current data aquisition and analysis hardware and software. Total system integration is through a personal computer based workcell controller. Segment inspection demands the use of a gantry robot for the EMAT/ET inspection system. The EMAT/ET system utilized similar mechanical compliancy and software logic to accomodate complex part geometries. EMAT provides volumetric inspection capability while eddy current is limited to surface and near surface inspection. Each aspect of the systems are applicable to other industries, such as, inspection of pressure vessels, weld inspection, and traditional ultrasonic inspection applications.

Background

Initial manufacture and subsequent refurbishment of the space shuttle Advanced Solid Rocket Motor (ASRM) demand precise inspection of the motor casings, both in the forging and segment phases of production, to preclude catastrophic failure. Robotic NDE inspection for case discontinuities was determined essential to achieve the program goal of ensuring overall case integrity. Two inspection points were identified in the ASRM cycle. The first inspection would identify surface fissures created during the forging and heat treatment manufacturing steps. Detection of flaws at this stage prevents scraping components downstream in the process. Inspection of segments (assembled forgings) during initial manufacture and refurbishment is the second point, and occurs early in the production cycle to ensure new segment integrity and check for cracks propagated during splashdown.

The sizes and complexity of forgings and segments requires the use of advanced robotic techniques for automated inspection employing state-of-the-art NDE. The approach to each inspection was driven by unique functional requirements requiring different robotic and NDE methods. In both cases, Babcock & Wilcox, CIM Systems was contracted to provide the innovative and rugged solutions.

Eddy Current Inspection System

ASRM cylindrical forgings span a wide range of sizes and complexity of features. Inner diameters range from 142.495 inches to 149.625 inches and outer diameters from 145.180 inches to 161.750 inches. Heights range from 34.50 inches to 148.420 inches. The forgings can weigh as much as 15,000 lbs. As a result of heat treatment, component out-of-roundness (circularity) could be as much as seven inches. Features may be final machined or forged, requiring additional machining in subsequent processes. Complex geometries include radii, corners, chamfers, weld prep transitions, protrusions, flanges, and multi-axis curvatures. The critical flaw size was set at 0.125 inch long by 0.035 inch deep for all inspections.

A vertical five axis system comprising two, two axis manipulators and a rotary table (figure 1) controlled by an eight axis servo controller provides the necessary motion for forging inspection at the Babcock & Wilcox, Aerospace Components Division, the manufacturer of ASRM segments. Six, single channel eddy current (ET) instruments are used for surface and near-surface flaw detection. A personal computer based, workcell controller integrates the motion control, data acquisition, and

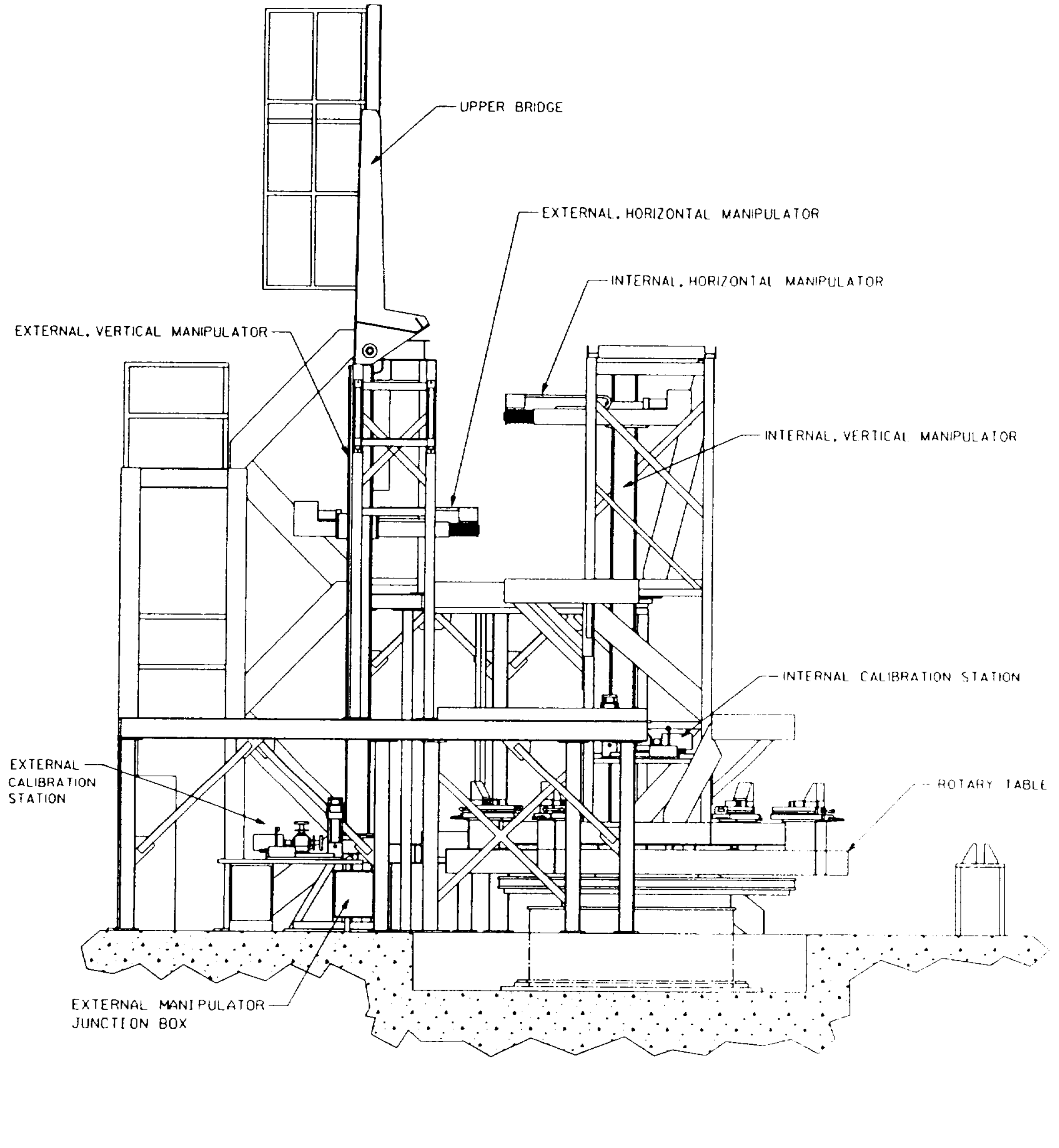

UPPER BRIDGE
EXTERNAL, HORIZONTAL MANIPULATOR
INTERNAL, HORIZONTAL MANIPULATOR
EXTERNAL, VERTICAL MANIPULATOR
INTERNAL, VERTICAL MANIPULATOR
INTERNAL CALIBRATION STATION
EXTERNAL CALIBRATION STATION
ROTARY TABLE
EXTERNAL MANIPULATOR JUNCTION BOX

Figure 1. Eddy Current Inspection System

horizontal axis for the positioning of over thirty pieces of unique eddy current tooling. Critical to this application is maintaining proper part contact. The unique solution incorporates four axes of compliancy: x, z, pitch, and yaw. Adaptive control is utilized on the x-axis compliancy to overcome part out-of-roundness. Sensory techniques and custom algorithms are employed on other compliancy axes to maintain part contact and detect possible collision.

Manipulators and Motion Control

Horizontal Axis Manipulators

The horizontal axis manipulators are custom designed extendable/retractable assemblies mounted on the vertical axis carriage for both the internal and external vertical axes (figure 2). Each is perpendicular to the vertical axis and mounted so that it is on a radial line of the rotary table. This is important so that the tooling mounted to the end of the horizontal axis is perpendicular to the surface of the ASRM part. Otherwise excessive vibration will occur.

The purpose of the horizontal axis is to position the end-of-arm tooling in towards or out from the surface of the ASRM part. Since ASRM parts are not perfect circles, a special subassembly is incorporated into the horizontal axis to compensate for the out-of-roundness. This subassembly is the linear, or x-axis, compliancy device, properly known as the compliancy device assembly (figure 3). This is mounted to the free end of the axis. At the end of the linear compliancy device is a flange mounting plate for mounting tooling components.

Extension and retraction of the horizontal axis is accomplished as follows. A fixed ball screw mounted concentrically within a large hollow shaft, with the ball nut secured to the rear of the shaft, allows the hollow shaft to extend and retract as the ball screw turns and the ball nut translates down the screw. Linear ball bearing bushings mounted in the forward end of the axis housing allow the manipulator to be subjected to large transverse loads without detriment. This feature, and space constraints, were the deciding factors in designing a custom horizontal axis versus using an off-the-shelf linear actuator. Transverse loads on the order of 300 pounds can be applied. Off-the-shelf actuators can typically withstand ten percent of their axial load, maximum, as a transverse load, making them inefficient for transverse load applications. Cam followers are used to eliminate any rotational movement of the hollow shaft and ball screw combination due to overhanging loads off the axis.

The fixed end of the ball screw at the rear of the axis housing is driven in parallel using a positive power transmission belt and sprockets. A servo motor fitted with an absolute encoder provides the necessary driving rotary motion and position feedback. The axis is capable of 108 inches per minute with a total stroke of 35 inches between soft limits.

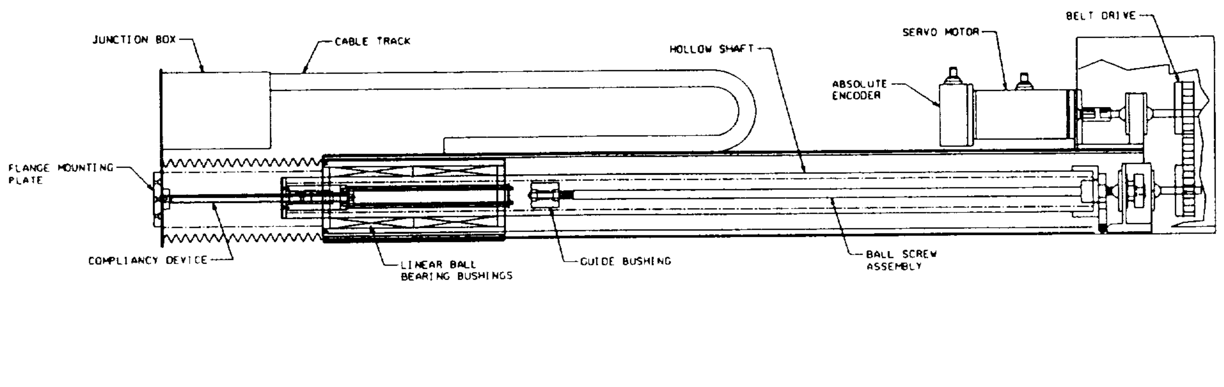


Figure 2. Horizontal Axis Manipulator.

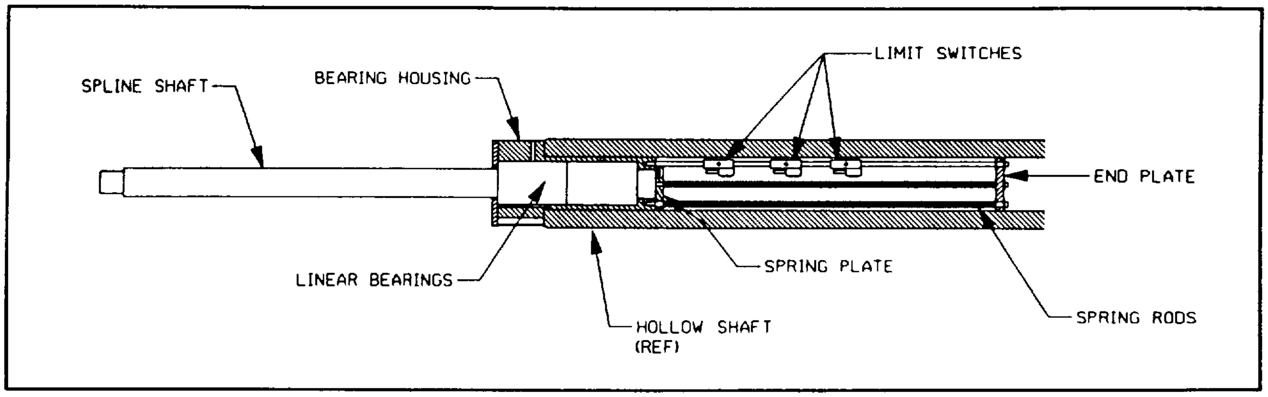


Figure 3. Compliancy Device Assembly

Compliancy Device Assembly

As stated above, this assembly is mounted into the hollow shaft in the end that extends and retracts. The design utilizes a ball spline assembly for smooth linear motion, transverse load capacity, and constraint from rotational motion. The ball spline ball bearing bushings are mounted in a housing fixed within the end of the hollow shaft. The spline shaft end within the hollow shaft is mounted to a plate and spring combination which provides sufficient spring force to push tooling and sensors against the ASRM component maintaining sensor-to-part contact. Three sensors mounted to the compliancy device assembly within the hollow shaft are used to detect the location of the end of the shaft. If the ASRM component forces the shaft inward, the rear sensor is actuated causing the motion controller to retract the horizontal axis to the center of the compliancy device. If the ASRM component moves away causing the spline shaft to extend, the forward sensor will be actuated causing the motion controller to extend the horizontal axis.

This combination of limited travel, spring resistant compliancy, sensor feedback, and control permits horizontal axis compliancy within the entire stroke of the axis, 35 inches.

Arm Compliancy Device Assembly

Though actually categorized as part of the end-of-arm tooling, the arm compliancy device assembly (figure 4)is discussed here because the item remains attached to the end of the horizontal axis for all inspections. It is mounted to the end-of-arm mounting flange. Attached to this assembly is a integrated, quick change tooling dovetail which allows for rapid tool changes, alignment, and rigidity. Where the compliancy device assembly discussed above provides x-axis compliancy, this device provides z-axis and rotational compliancy.

A linear cross roller bearing assembly is mounted vertically between the base plate and an intermediate plate. Flat air cylinders are used at each end of bearing travel to provide the necessary spring force to allow resisted vertical motion of attached tooling. Independent regulated air pressure to each cylinder permits counteracting gravitational forces and aligning the tooling.

Springs sandwiched between the intermediate plate and the final tooling mounting plate allow for pivotal compliancy about two perpendicular axes.

Embedded limit sensors on the z compliancy and the two rotation compliancy axes are used to detect abnormal operational conditions such as a collision between tooling and an ASRM component.

Though these features are available as off-the-shelf components, space constraints dictated a custom design. In addition to the compact design, more freedom of movement for compliancy is provided than available from off-the-shelf component vendors.

Vertical Axis Manipulator

There are two vertical axis manipulators, internal and external (figure 4).

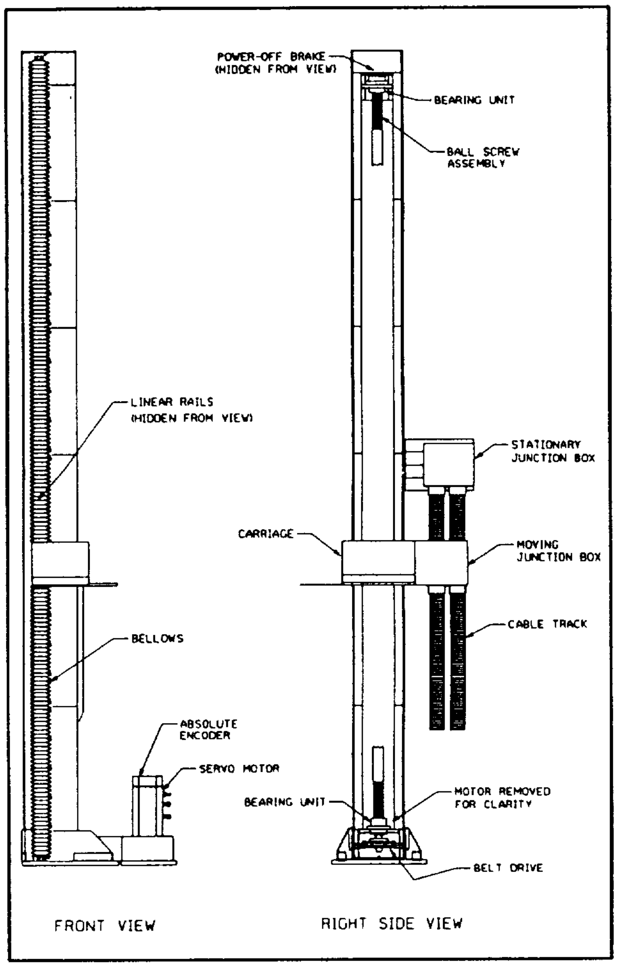


Figure 4. Vertical Manipulator Assembly

A wide flange I-beam serves as the main support structure. A linear rail fitted with two linear bearings is mounted to each flange. The vertical carriage is bolted front and back to the front and back pairs of linear bearings. The ball nut of the vertical axis ball screw is mounted rigidly to the carriage. The ball screw is mounted between the beam flanges at the top and bottom using four row angular contact bearing assemblies. A power-off DC brake is spline coupled to the top of the ball screw to prevent back driving. The bottom of the screw is the drive end.

Rotation of the ball screw is through indirect 3:1 power transmission belt drive. Matched sprockets attached to the ball screw and motor shaft achieve the 3:1 gear ratio. As the ball screw is rotated, the ball nut and thus, the carriage, travel up and down the beam.

The motor is a AC brushless servo fitted with an absolute encoder for closed-loop position control. The drive allows for a maximum axis velocity of 54 inches per minute. Travel between soft limits is 157 inches.

Rotary Table

The rotary table, manufactured by Koike Aronson, Inc. , is 120 inches in diameter and capable of 0-5 RPM rotation in both directions. The center of rotation of the table (approximated as the center of the table top) defines the center of the workcell. The table is rated at 20,000 lbs capacity though is easily capable of handling 30,000 lbs.

The drive train is the geared main bearing (slew ring) driven by two pinion gears. Two gear boxes along with the gearing of the main bearing and gear ratio of the power transmission belt drive provide a 297:1 gear ratio. An DC servo motor fitted with an incremental encoder is the prime mover. A belt drive is used between the motor and input shaft of the gear boxes.

A second encoder is mounted to provide 1:1 feedback of the rotary table top position.

Motion Control

All motion of the ET Inspection system originates from the part program stored in the WorkCell controller. The program is down loaded to the motion controller via RS-232 were it is stored. The part program in association with the motion control hardware performs all closed loop motion control, I/O control and fault handling and recovery, see system block diagram figure 5.

The motion control system consists of five axes of motion. These axes include two (2) for the internal manipulator, two (2) for the external manipulator and one (1) for the rotary table. A sixth open loop feedback only axis is used on the rotary table for homing and position display. Each axis is controlled via a dedicated controller card located in the motion controller . The card communicates with the associated drive via an analog signal and receives feedback information from the axis encoder.

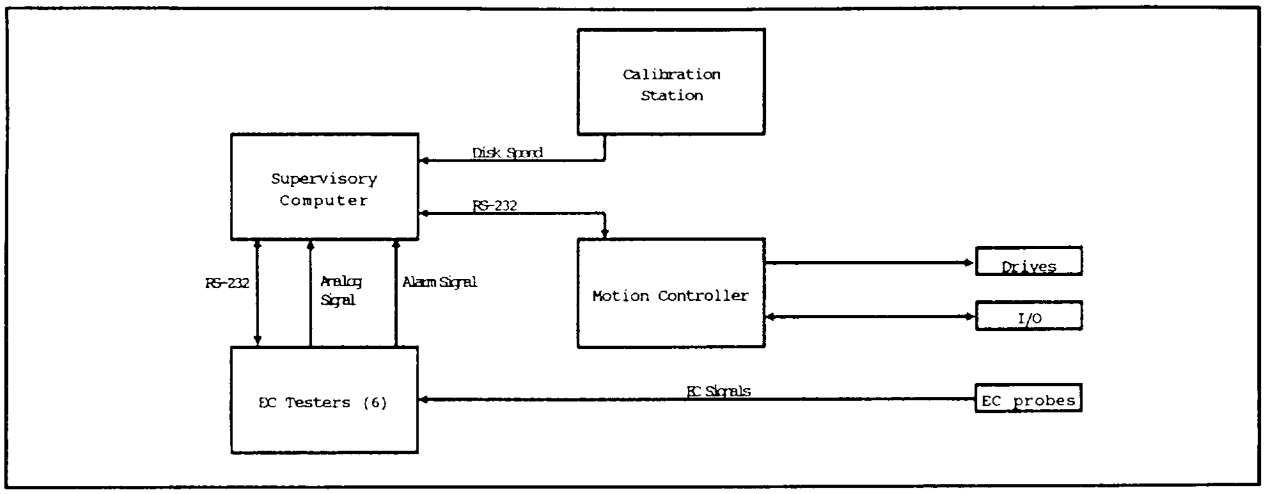


Figure 6. Eddy Current Inspection System, System Block Diagram

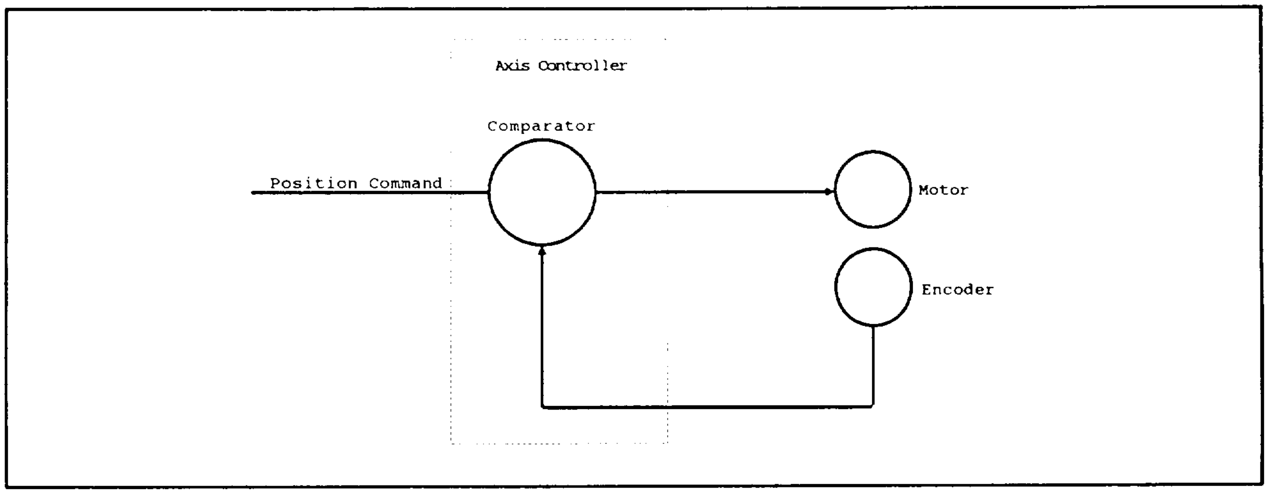


Figure 5. Feedback Loop.

Each axis of the ET Inspection system consists of a servo motor and drive amplifier. The drive accepts analog voltages from the axis controller cards, and in turn, commands the axis motor to rotate, producing axis motion.

The controller card is responsible for accepting motion commands from the motion control program and correlating this with the feedback signal (encoder) to produce an output com d to the axis drive, figure 6.

The system incorporates two types of encoders, incremental and absolute. Each manipulator axis consists of an absolute encoder with a 1024 line count. The rotary table axis utilizes two incremental encoders, one for closed loop control of the axis and the other for homing and position display functions.

Tooling

Tooling is provided to hold the eddy current probes for positioning by the manipulators. For each ASRM part feature such as a T-stiffener edge, there is a tooling setup. All setups include the tooling extension tube and end-of-arm tooling base block. The base block is used to hold the membrane eddy current probe and also serves as an attachment base for special feature tooling. Special feature tooling is provided to hold all other eddy current probes.

The base block includes two eddy current proximity probes used to detect the surface of the ASRM parts. These sense the presence of the metal surface. If the ASRM part surface is within the sensing range, it is known that the tooling wheels are contacting the part surface. The importance is that the part surface is used as a reference for positioning the tooling to ensure the eddy current probe is on the part surface to take data. If the part surface is not within the sensing range of the proximity sensors, the switch signal indicates a sensor liftoff fault.

Sensors, Data Acquisition and Analysis

Each of the automated NDE systems provided by CIM systems for the inspection of rocket motors consisted of Eddy Current (EC) probes. Eddy Current Test (ET) is the primary inspection method on the ET Inspection system. Due to many complex features located on the rocket motor forgings, over 30 specially designed probes and probe fixtures are needed to inspect geometries such as T-stiffeners, weld joints, chamfers and radii. Each fixture consists of a quick release mechanism to quickly provide for EC probe changes, thus a fixture may be used for many probes, see figure 7. The probe itself not only consists of the EC coils but also provides for methods of maintaining lift-off to the inspection surface. Though many compliancy devices are provided in the system to provide for part out-of-roundness, the first two defenses for maintaining part contact are on the probe itself. A minimum of three adjustable wear pins are provided on the surface of the probe to provide a static lift-off to the rocket motor. The probe body attaches to the fixture interface plate via 3 or 4 shafts encircled with springs. The springs allow the probe to float and mimic forging movements.

Each probe consist of six (6) differential coil arrays. Each coil is staggered from the previous coil with 25% overlap to assure no flaws will pass between coils. Each coil, or channel, has a dedicated Nortec 19e EC tester. Each tester serially interfaces to the supervisory computer which allows for remote setting of inspection parameters such as alarm thresholds, signal gain, inspection frequencies and a variety of others. Inspection parameters are developed as a part of the calibration process. Each probe must be calibrated prior to conducting inspection of the rocket motor. The calibration consists of simulating the inspection surface

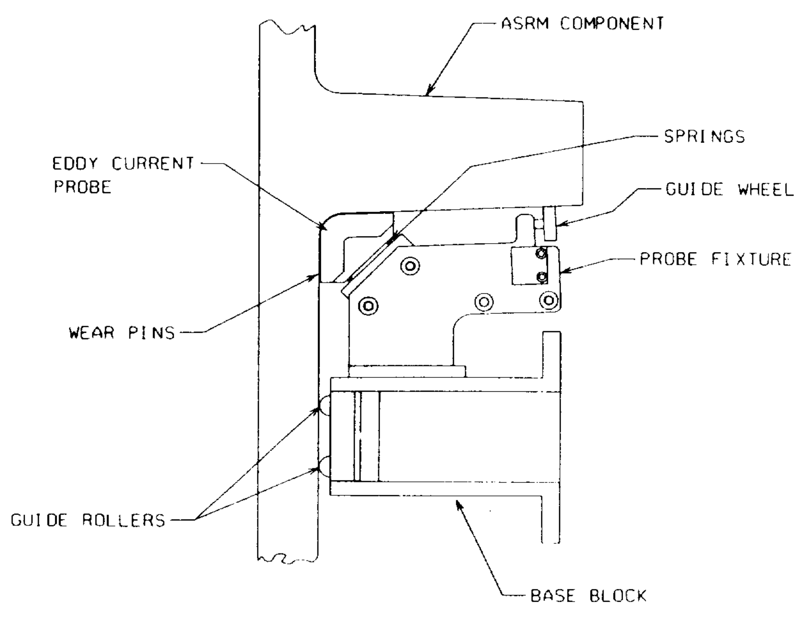


Figure 7. Eddy Current Probe, Fixture, and Base Block

on a disk of identical material, with six (6) EDM notches of half the critical flaw size (.125 X .035). The probe is positioned on the disk as the disk is spun at a surface speed identical to the intended inspection speed. Parameters are then adjusted on the tester to maximize the signal to noise ratio (S/N) on each channel.

Data acquisition is accomplished via an analog output supplied by each of the testers. The analog signal is digitized through a Analog Devices Analog to Digital (A to D) converter card located in the PC backplane. The card digitizes each of the channels at a rate of 8.3 KHz with each sample point consisting of 16 bits of information. The data is buffered on the card and is retrieved on an as needed basis. The data is retrieved and stored to disk based on an alarm signal generated by the tester. The tester will generate a discrete output if the EC signal brakes the alarm threshold set during the calibration procedure. Based on the alarm signal 1 inch of data before and after the alarm is retrieved and stored to disk for analysis and archiving.

Due to the massive size of the inspection parts relative to the critical flaw size, position of the EC probe must be monitored, retrieved and stored with the inspection data upon an alarm condition. The inspection piece being a cylinder, two coordinates must be known to relocate a flaw indication, Z-axis of the manipulator and Θ-axis of the rotary table. The rocket motor is inspected in bands and therefore position of the horizontal axis is static throughout the band. This position is relayed to the supervisory computer from the motion control subsystem and is retained for further processing in the event of an alarm condition. An encoder is used to track the position of the rotary table. Quadrature TTL level pulses are produced by a 4096 line count incremental encoder. These pulses are collected and processed by the supervisory computer.

Inspection Process

The main operator interface of the ET Inspection system is the supervisory computer. The supervisory computer provides for the development of the inspection plan or the inspection recipe. The inspection plan is developed by the NDE engineer and is a step by step sequential operation similar to CNC code. The plan, which can be developed off-line, is the road map for the inspection process of a forging. It provides the motion control system with motion variables, the EC testers with inspection parameters and other house keeping functions such as operators name and ID number, part identification number, probe numbers and data file names.
Key words are used to represent inspection functions. For example, the key word "EXTERNAL SCAN" and the associated parameters perform an external inspection of the forging.

```
EXTERNAL SCAN 5.0 23.00 75.00 .50 34.00
```

EXTERNAL SCAN is the key word for an external inspection. The 5.0 represents the rotational speed of the rotary table in RPMs. 23.00" is the starting Z position of the external manipulator and 75.00" is the stopping position. The increment amount of .50" translates to 104 bands of inspection. The final parameter represents the static X position of the external manipulator for the inspection. Other key words include TOOLCHANGE [position][tool ID], DOWNLOAD [file], SHUTDOWN, UPLOAD [file], MOVE [axes][coordinates] and CAL [internal or external][disk speed]. A complete inspection plan for conducting two scans may appear as follows.

```
CAL EXTERNAL 230
UPLOAD xyz.par
TOOLCHANGE lower xyz
EXTERNAL SCAN 5.0 23.00 75.00 .50 34.00
CAL EXTERNAL 230


CAL EXTERNAL 230
UPLOAD abc.par
TOOLCHANGE lower abc
EXTERNAL SCAN 5.0 75.00 85.00 .50 34.00
CAL EXTERNAL 230
```

This inspection plan first conducts a calibration of probe xyz. The disk speed of 230 RPM will translate to the same surface speed of the forging at 5 RPM. Once the calibration is complete and all EC channels respond equally with sufficient S/N the parameters are uploaded and stored for documentation and verification purposes. The external manipulator is then positioned to the lower tool change position and the operator prompted to install tool xyz. An acknowledgement is made that the tool has been installed and the manipulator will position itself to the starting location of 23.00" in Z and 34.00" in X. The operator is then allowed to make any fine positioning if necessary and the inspection will begin. A calibration is again conducted once the scan is completed to once again verify the functionality of the EC probe. The second scan is similar to the first, only at a different part of the forging. Once the inspection plan is complete a report is generated with the results of the inspection. The report includes all flaw indication locations and house keeping information.

EMAT Inspection System

Segments are multiple forgings welded end-to-end. A minimum of three full size forgings, such as weld-weld or field-weld forgings are welded together. Or, for more complex segments, full size forgings are connected to multiple specialty forgings, such as T-stiffener and IETA forgings. Segments are approximately 45 feet high. In addition to the features encountered during forging inspection, segments have threaded and through holes requiring inspection. All features are in the final

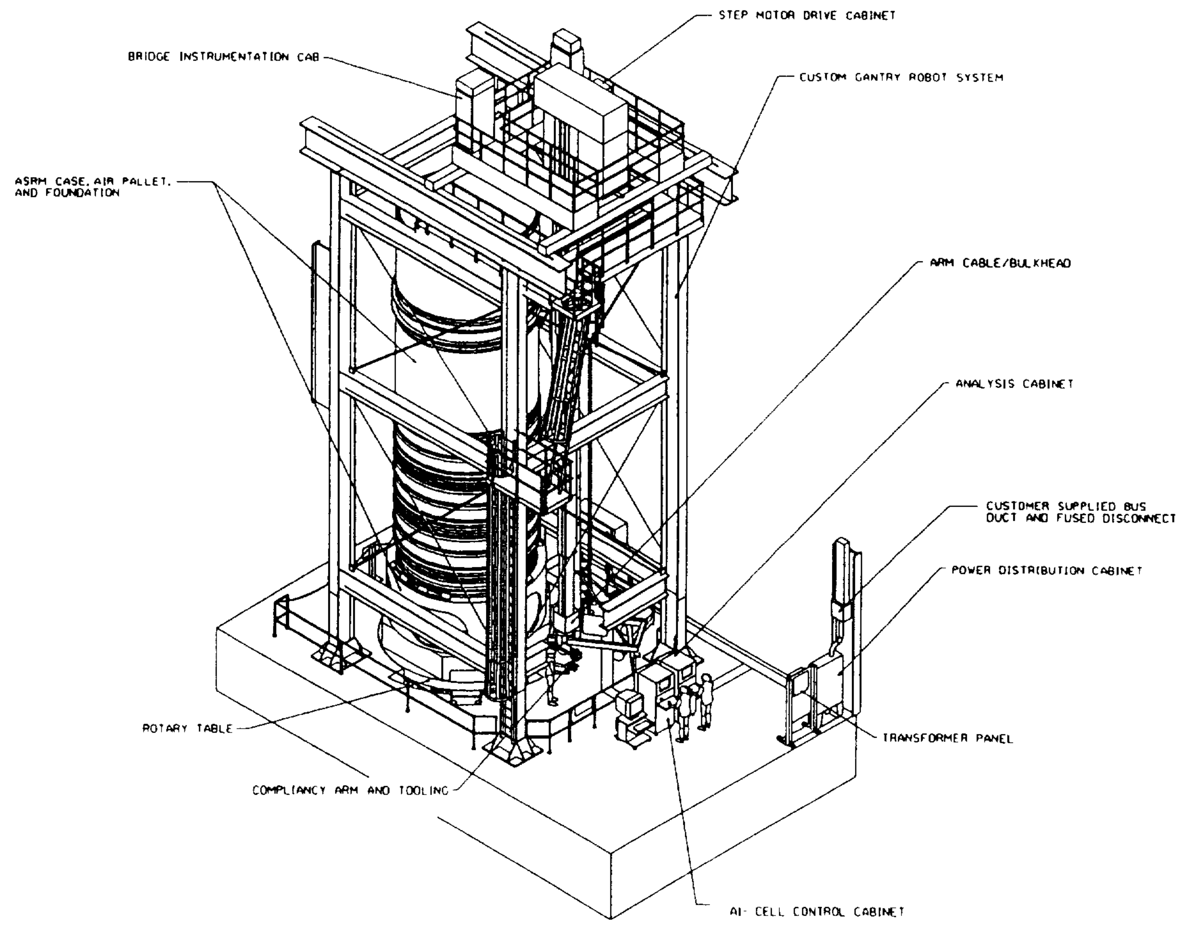

STEP MOTOR DRIVE CABINET
BRIDGE INSTRUMENTATION CAB
CUSTOM GANTRY ROBOT SYSTEM
ASRM CASE, AIR PALLET, AND FOUNDATION
ARM CABLE/BULKHEAD
ANALYSIS CABINET
CUSTOMER SUPPLIED BUS DUCT AND FUSED DISCONNECT
POWER DISTRIBUTION CABINET
TRANSFORMER PANEL
ROTARY TABLE
COMPLIANCY ARM AND TOOLING
AI- CELL CONTROL CABINET

Figure 8. EMAT/ET Inspection System.

machined state, although in the refurbishment process irregularities may be severe due to the intense heat generated during liftoff and the shock of splashdown.

Volumetric inspection of segments was specified for the assembly and refurbishment facility in Iuka, Mississippi prior to assembly into ASRM's. Electromagnetic acoustic transducer (EMAT) was selected as the primary NDE technique with ET being used on limited special features. EMAT is similar to UT in functionality except no couplant is required. The sound is produced by an electromagnetic acoustic interaction within the material, which facilitates high speed automated inspection. Complete, 100% inspection of a single segment takes approximately 10 hours to perform, a significant improvement over the current manual process.

In order to inspect the tall segments, a four axis gantry robot is integrated into a five axis robotic workcell (figure 8). Integrated through a UNIX based workcell computer are the robotic and data acquisition systems, each having a dedicated controller. Mechanical compliancy is also critical to this application and is a refinement of that used for the forging inspection station. The system features:

- A 44 foot telescoping mast
- 5 axes of coordinated motion
- CNC part programming
- Fully automated inspection techniques
- Advanced data acquisition and analysis capabilities, including A, B, and C scans
- A multi-tasking operator interface

Robot & Control

The Electromagnetic Acoustic Transducer/Eddy Current Test (EMAT/ET) system is responsible for inspecting new and refurbished rocket motor segments. The segments vary slightly in size and are approximately 45' in height and 12' in diameter. Similar to the ET Inspection system the rocket motor is placed on a rotary table and rotated while the test probes are indexed over the surface. Because of the increase in size of the inspection piece, a robotic gantry system was chosen over manipulators, see figure 8. The robot possesses an X bridge assembly, a Z mast assembly and a two axis wrist. The result is a 4 axis robot, X, Z, θ_1, θ_2, capable of reaching and inspecting 100% of the rocket motor. The gantry spans 65' from the floor to the top of the bridge with 44' of stroke on the Z axis. Each axis, including the rotary table, is coordinated and controlled by the CIMROC 4000x robotic controller, see system block diagram figure 9. The controllers functions are to perform closed loop servo control, communicate to the supervisory computer and perform system I/O.

The supervisory computer consists of a Hewlett-Packard 720 workstation running under UNIX and functions as the main operator interface and provides the platform for running the application software. A user-friendly menu system allows the operator to develop scan

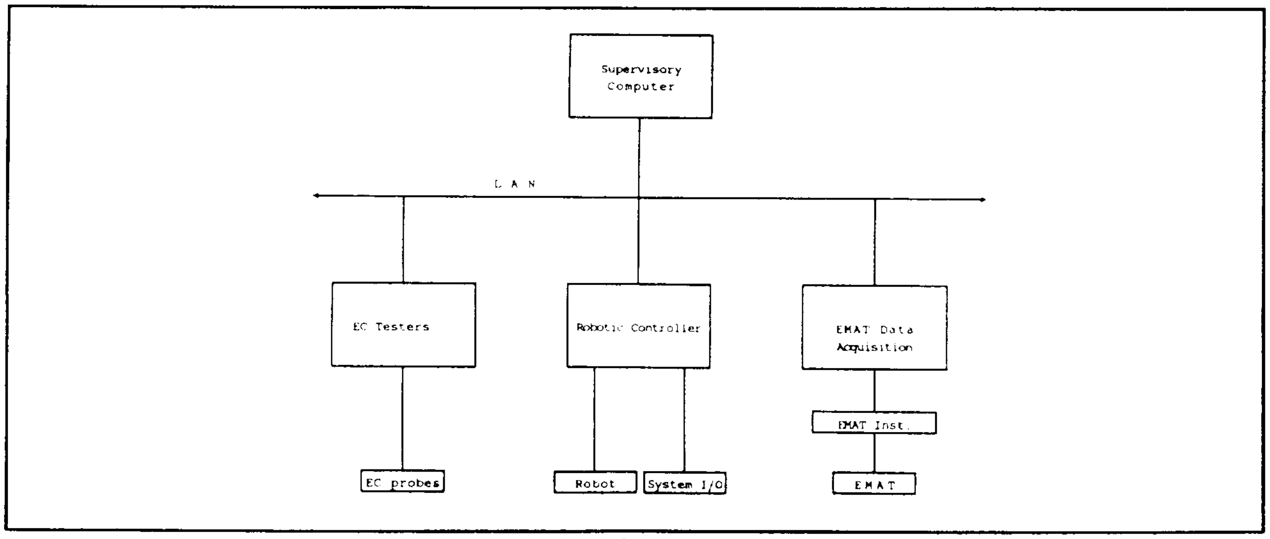


Figure 9. EMAT/ET System Block Diagram.

plans, run inspections, download motion control functions, perform diagnostics and analyze data. An X-terminal is supplied to allow data analyzing as inspections are being run.

Compliancy Arm

A compliancy arm was designed and manufactured for use in the EMAT/ET Inspection System in order to compensate for uncertainties similar to that found with the Eddy Current Inspection System. The arm incorporates all features found in the group of compliancy features, previously discussed.

There are three sections to the compliancy arm: X compliancy, Z compliancy, and pitch/yaw compliancy (figure 11). Each section relies on computer adjustable, regulated air pressure to set the desired spring rate of the compliancy. This provides compensation for varying loads. Switches are used for over travel detection on the z and pitch/yaw compliancy axes. The axis compliancy switches are used for adaptive control similar to the linear compliancy of the eddy current inspection system.

Tooling

Tooling for the EMAT/ET Inspection Systems consists of a dual, rotating EMAT, single, rotating EMAT, through hole EC tooling assembly, and EC probes for manual inspection. More than 90% of a segment's volume can be inspected using either the dual or single, rotating EMAT tooling assemblies.

Development of the dual, rotating EMAT was driven by the throughput requirement of 10 hours per segment, average. An EMAT probe can detect anomalies oriented within $\pm 2.5°$ of the EMAT scanbeam. (Some tests have validated scanning for indications within $\pm 10°$.) The scanbeam covers a forward and aft surface region that is generally trapezoidal. By using two EMAT probes mechanically coupled to rotate to the same angular

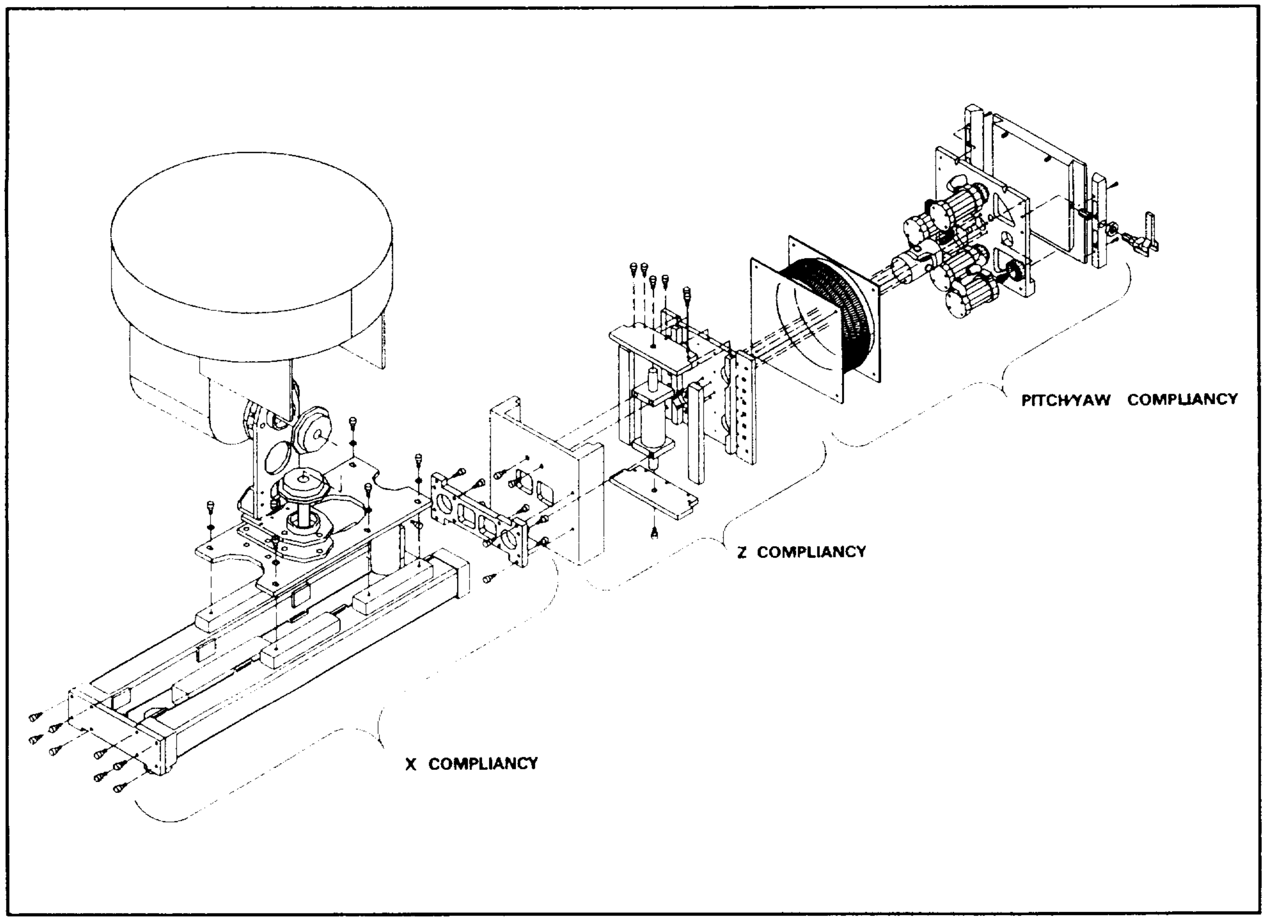


Figure 10. EMAT/ET Compliancy Arm.

orientation and controlling overlap to cover the deadzones of the probes, an optimal scan configuration is achieved (figure 11). For each angular orientation, there is a preferred center-to-center spacing for the two probes. At lower angular orientations, the volume that can be scanned with one pass of the tooling is doubled, there is no dead zone to cover. At higher angles, approximately 150% of the volume that can be covered by one EMAT probe is scanned with the dual EMAT tooling. The greatest improvement to scanning efficiency is the minimization of increment overlap, i.e., incrementing the tooling vertically. Without the dual EMAT tooling, dead zones would be covered by increment overlap which would reduce the effectiveness of one EMAT probe to less than 50%.

Step motors are used to position the dual EMAT probes both angularly and translationally, with respect to each other. Each EMAT probe is mounted with a double set of pivot points, allowing the EMAT probe to conform to the surface. Both probes are connected using a structural tube; however, the secondary probe is connected to the tube through a linear cross roller bearing which permits linear movement towards and away from the primary EMAT. A ball screw driven by one of the step motors controls the position of the secondary EMAT. A large radial bearing connects the tube to the tooling mount plate, the connection point to the robot arm. Concentric with the bearing is a spur gear driven by a mating pinion gear. The pinion gear is driven through a planetary gear box by the other step motor.

Single EMAT inspection is required in domed areas due to the complex curvature. The dual EMAT does not allow the flexibility necessary for domes because constraints between the two probes of the dual EMAT are employed to guarantee coverage between the two probes. The single EMAT tooling incorporates the rotation and translating features of the dual EMAT. Like the dual EMAT, the rotation of the single EMAT is used to adjust the probe to different orientations for randomly oriented volumetric indications. The translational feature of the single EMAT is used to scan the probe across small surfaces in lieu of using the motion capabilities of the robot. One example is scanning the vertical section of a T-stiffener ring. Step motors are used to drive the rotation and the translation of the single EMAT tooling.

A through hole, eddy current probe is fixtured in tooling to scan the many bolt holes found in the flanges of segments. The probe has one eddy current sensor coil that is spring loaded. The probe is plunged into a through hole at a known rate while rotating. This combination is necessary to ensure proper overlap to guarantee 100% coverage.

Sensors, Data Acquisition and Analysis

The EMAT/ET system not only must conduct surface inspection but also provide for a volumetric examination. Unlike ET, Ultrasonic Testing (UT) with EMATs is now coming into its own with Babcock & Wilcox leading the way. An EMAT consists of a coil of wire and a magnet. A strong field is produced at the surface of the material by the permanent magnet. This produces an electromagnetic interaction within the material resulting in sound waves being generated. If voids such as cracks are encountered within the conductor, the wave is reflected and sensed by the receiver.

The Accusonex data acquisition system provides for data collection and analysis for the EMAT inspections. The system consists of an HP-V382 embedded controller that performs functions such as pulse control, signal digitization and serves as an interface between real-time data acquisition and data storage to disk. The system pulses the EMAT magnet and intern receives an analog signal from the EMAT instrumentation. The signal is digitized, buffered and presented to the operator in real-time in the form of A, B, or C-scans. All data, including ET, is stored to optical disk for analysis and archiving.

EC probes supplement the inspection in areas too small for EMAT. These areas include T-stiffener radii, bolt holes and other miscellaneous geometries. The ET instrumentation (MIZ-30) drives up to eight (8) inspection coils that are scanned over the rocket motor. The coils induce current into the rocket motor and sense changes in the electrical characteristics of the material. The electrical signal is digitized in the MIZ-30 and is transferred over a Local Area Network (LAN) to the supervisory computer for storage. The operator is presented with the data in real-time in two forms, amplitude vs. time and amplitude vs. phase.

Technology Applications

Technologies from both systems have been implemented in other applications, particularly the robotic tooling concepts and EMAT based NDE technology. These techniques can be utilized in automated inspection of pressure vessels and other components requiring sophisticated NDE inspection to ensure part integrity.

EMAT is useful in applications requiring the output achieved from ultrasonic inspection. However, EMAT has the advantage that no couplant is required to carry the signal.

For more information, contact Glenn E. McNeelege at (804)948-1347 or Chris Sarantos at (804)948-1348.

4061 **AIAA-94-1223-CP**

AUTOMATION FOR NONDESTRUCTIVE INSPECTION OF AIRCRAFT

N94-30572

M. W. Siegel*
Carnegie Mellon University
Pittsburgh PA 15213-3891

[illegible]

Abstract

We discuss the motivation and an architectural framework for using small mobile robots as automated aids to operators of nondestructive inspection (NDI) equipment. We review the need for aircraft skin inspection, and identify the constraints in commercial airlines operations that make small mobile robots the most attractive alternative for automated aids for NDI procedures. We describe the design and performance of the robot (ANDI) that we designed, built, and are testing for deployment of eddy current probes in prescribed commercial aircraft inspections. We discuss recent work aimed at also providing robotic aids for visual inspection.

I. Background

Our goal is to replicate and enhance the capability of aircraft skin inspectors who use hand-held instruments (and their own senses and intelligence) to detect and classify flaws in aging aircraft. Our underlying concept is to use mobile robots, automated control, and automated interpretation of sensors and instruments to make *difficult measurements in difficult environments*. Potential application area include not only airplane skins, the subject of this paper, but also problems such as bombs in luggage, contraband in cargo containers, verification of disarmament treaty compliance, characterizing environmentally contaminated sites, and a variety of manufacturing problems, e.g., measuring composition gradients in large process tanks, transportation problems, e.g., bridge inspection, and scientific research problems, e.g., checking the integrity, alignment, etc, of large instruments such as radio telescopes and particle accelerators. These few examples just begin to suggest the universe of potential application areas and specific applications. A general hierarchical paradigm for organizing the common issues of *measurement*, *manipulation*, *mobility*, and *monitoring* characteristic of all these problems is illustrated explicitly for the aging aircraft problem in Figure 1.

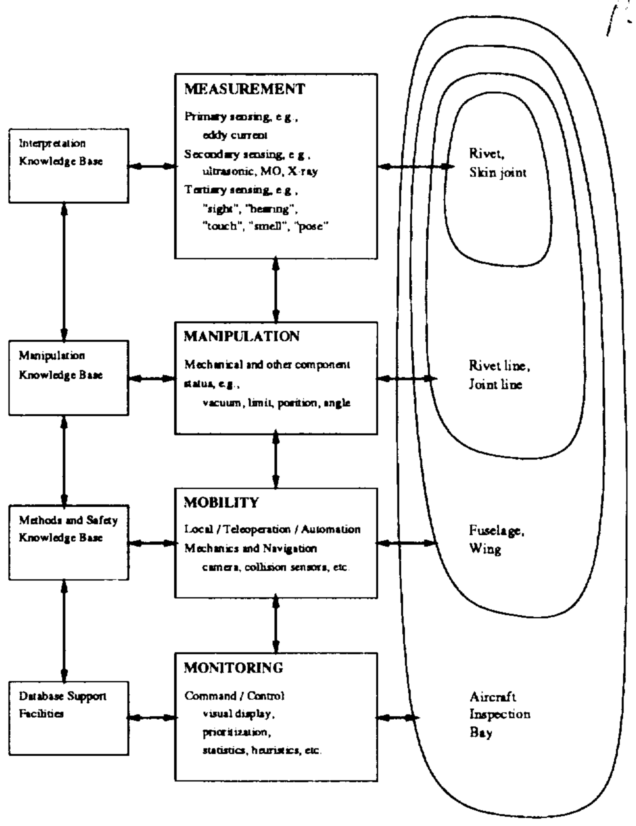

Interpretation Knowledge Base
MEASUREMENT
Primary sensing, e.g., eddy current
Secondary sensing, e.g., ultrasonic, MO, X-ray
Tertiary sensing, e.g., "sight", "hearing", "touch", "smell", "pose"
Rivet, Skin joint
Manipulation Knowledge Base
MANIPULATION
Mechanical and other component status, e.g., vacuum, limit, position, angle
Rivet line, Joint line
Methods and Safety Knowledge Base
MOBILITY
Local / Teleoperation / Automation
Mechanics and Navigation camera, collision sensors, etc.
Fuselage, Wing
Database Support Facilities
MONITORING
Command / Control visual display, prioritization, statistics, heuristics, etc.
Aircraft Inspection Bay

Figure 1: The 4M-s of automation for aircraft inspection.

II. Inspection of Aging Aircraft

Aircraft skins inflate and deflate with each cycle of pressurization and depressurization. The resulting stress causes several kinds of damage, primarily radial cracks around rivets, delamination of skin joints, and subsurface cracks in the structural members to which the skin is attached. Delamination is exacerbated by corrosion, which is particularly prevalent in warm moist climates. Cracks and corrosion, accelerated by island-hopping operation, resulted in April 1988 in a large section of skin tearing off the top of the fuselage of an Aloha Airlines Boeing 737. The resulting press coverage of the

*Senior Research Scientist, The Robotics Institute, School of Computer Science airplane's seemingly miraculous safe landing brought these problems prominently to the attention of the public, and resulted in an aggressive prevention, detection, and remediation program by aircraft operators in close cooperation with each other, the aircraft manufacturers, and the FAA[1, 2, 3]. Structural effects of aging in other areas, such as engines, fuel tanks, landing gear, etc, possibly will be the subjects of future automation research, but for the present our program is concentrating on skin and the immediate supporting substructures.

Through programs of periodic inspection of known problem areas on each aircraft type, skin cracks and corrosion are typically found well before they reach hazardous size. The problem areas are specified by "service bulletins" issued by aircraft manufacturers, and by "airworthiness directives" issued by the FAA. Compliance with airworthiness directives is mandatory. Compliance with service bulletins is at the airline operators discretion, but we are told that in practice they are treated as mandatory.

About 90% of skin inspection is visual, by inspectors trained for the task, most of the remainder is by eddy current probes, and a fraction of a percent is by other instrumentation of which the best known is probably ultrasonic. Our program is focused in its initial phases on automation as an aid to skin inspection using eddy current probes. Initially we will use machine vision to aid probe placement and robot navigation and to update the navigation database with descriptions of patches and other deviations from "as designed". We are beginning to investigate automated aids to visual inspection via a new program in which a small limited functionality robot will be used deploy 3D-stereoscopic cameras. Working with experienced visual inspectors, we will evaluate the acceptability of computer aided remote (teleoperated) visual inspection.

Eddy Current Inspection

The eddy current method[4] uses a transmitting coil and a receiving coil (they may physically be one coil) coupled electromagnetically through the metal under inspection. Eddy current probes vary in tip area from several square centimeters to about one square millimeter, obviously trading off decreasing areal coverage for increasing sensitivity to small flaws as the size decreases. Anomalies in the impedance that characterizes the coupling indicate cracks, corrosion thinning, and other flaws. Inspectors generally watch an x-y oscilloscope display whose x-axis represents the in-phase (resistive) part of the impedance and whose y-axis represents the quadrature-phase (inductive or capacitive) part of the impedance. Figure 2 illustrates a probe, and Figure 3 illustrates typical impedance plane signals. The inspectors compare patterns traced out on the screen when the probe is passed over a potential flaw with the pattern traced out when the same probe is passed over a calibration standard manufactured with a machined flaw in the simulated local structure. The probe geometry, operating frequency, scan path, etc, are chosen to optimize sensitivity to each anticipated flaw. High operating frequencies are attenuated in a short distance, and thus probe only the surface. Low operating frequencies penetrate deeper, and in some geometries can penetrate the skin entirely and probe for cracks in the supporting framework. Under typical operating conditions power levels are sufficiently low that the method is extremely linear, so it is possible to operate a probe with a composite multi-frequency transmitted waveform and to separate electronically the high-frequency surface-sensitive received signal components from the low-frequency substructure-sensitive components.

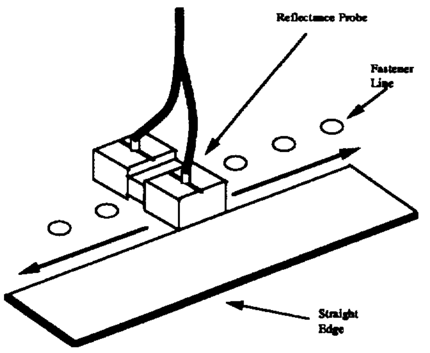


Figure 2: "Reflectance" or "pitch-catch" eddy current probe.

Modern eddy current systems can be set to alarm on traces that enter or fail to enter preset rectangular windows in complex impedance space. Initially we will rely on these alarms to alert the inspector to potential flaws indicated by anomalous signals. These areas will be marked for easy identification by the inspector, e.g., by daubing suspect rivet heads with a washable paint. Pattern recognition integrated with rule based systems is an accepted method for automating interpretation and classification of eddy current signals in other applications, e.g., inspection of heat exchanger tubes in nuclear power plants[5]. Neural network methods have been similarly successful in similar applications[6]. As the program progresses we will add additional software to implement promising approaches to automated and improved eddy current signal interpretation and classification.

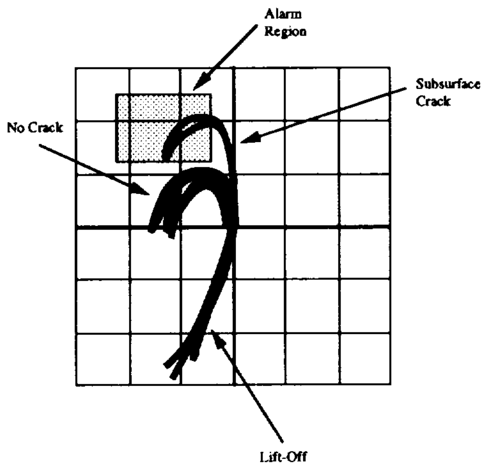


Figure 3: Eddy current signals in the complex impedance plane.

III. Automated NonDestructive Inspector

We considered many approaches to automation-assisted eddy current probe deployment, with three primary variants: the gantry-based "car wash", the vehicle-based "cherry picker", and the self-contained "window washer". The pros and cons of these alternatives have been discussed in detail elsewhere[7, 8, 9, 10, 11, 12, 13, 14, 15]; in summary, the "window washer" design that we eventually chose for the system is dictated by the pragmatics of fitting NDI nondisruptively into the flow of passenger aircraft maintenance operations. These constraints suggest a small (under one meter maximum dimension) mobile platform that is able to walk or crawl over most if not all of the aircraft skin, whatever its orientation. This capability we achieve with active (vacuum assisted) suction cups. A concept sketch for the resulting robot (ANDI, the Automated NonDestructive Inspector) is shown in Figure 4. It is not the easiest approach, but it is the most acceptable, and incidentally it is the approach that requires the most interesting enabling research.

Cruciform Design

Because there is generally more fore-aft than circumferential inspection path, the robot is designed with a cruciform geometry that enables it to move along fore-aft paths most rapidly; this results in a design in which it moves on circumferential paths somewhat more slowly, and in skew directions adequately, but a bit awkwardly. The mechanical design is sketched in Figure 5 and shown close-up (with eddy current probe in the foreground and a graphical depiction of the probe output on the computer

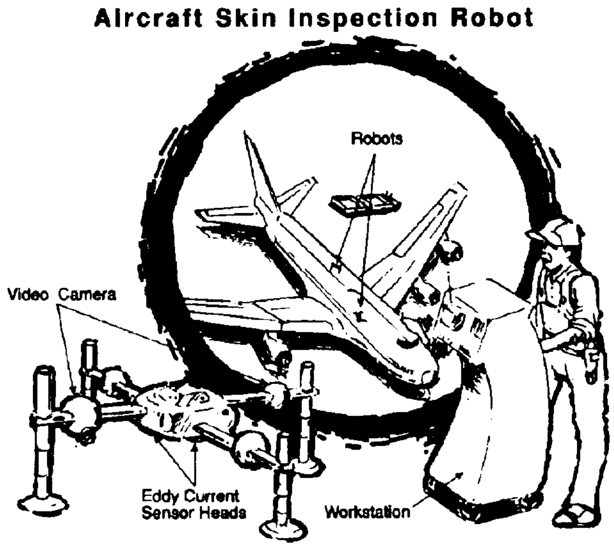


Figure 4: Concept sketch for ANDI.

monitor screen in the background) in Figure 6. It has many features in common with the class of mobile robots known in the literature as "beam walkers"[16, 17]. However unlike most beam walkers our robot is able to side step almost as easily as it can walk forward or backward. The two cross members ("bridges") are normally locked at right angles to the main longitudinal member ("spine"), but they can be released to pivot freely by about 15° in either direction; this permits the robot to steer and thus to travel along paths that are neither strictly fore-aft nor strictly circumferential. Pneumatically actuated up-down degrees of freedom on the four suction cups at the ends of the bridges enable the walking motion, and another pneumatic actuator enables the raising and lowering of the eddy current probe. The sliding motions of the bridges along and perpendicular to the spine are actuated by electric motors.

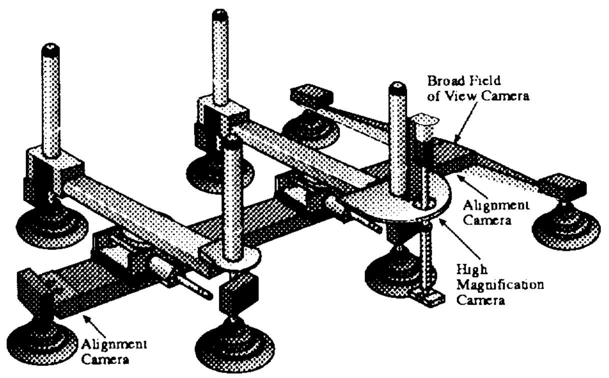


Figure 5: Mechanical features of ANDI, showing four camera mounting points.

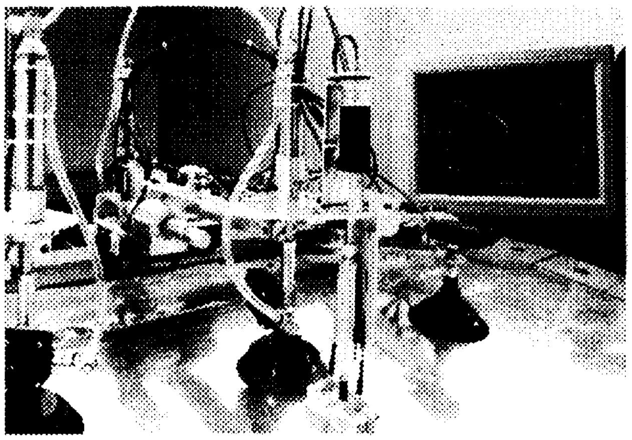


Figure 6: ANDI, showing eddy current probe and its signal on the computer monitor.

Alternative Designs

It is regrettably easy to confuse ANDI, particularly given its multiply anthropomorphic name**, with the much larger and more complex system of which it is essentially just the mechanical end effector. It is thus appropriate to emphasize explicitly that ANDI is just the first prototype mechanical end effector of a large and complex system (most of which is black boxes full of electronics and computers) that can accommodate many different end effectors. ANDI is designed to demonstrate the feasibility of using robots to assist inspectors of aging aircraft. But ANDI is not the last end effector that will ever be needed for this task. There are places on an airplane skin where ANDI cannot adhere, e.g., sharply curved regions around the nose, tail, and leading and trailing edges of the wings and horizontal and vertical stabilizers. There are places where ANDI can adhere but may turn out to be insufficiently agile to deploy the eddy current sensor in an effective pattern, e.g., perhaps around doors, windows, repair patches, etc. Our goal is to demonstrate what ANDI can do. We guess it can do something like 80% of the mandated and recommended eddy current inspections on DC-9 or Boeing 737 and larger aircraft. If ANDI proves its technical and economic worth in these applications, we are confident that we (and others!) will be able to design as many specialized mechanical end effectors as are needed to cover the applications ANDI cannot.

A block diagram of the currently envisioned complete system is shown in Figure 7.

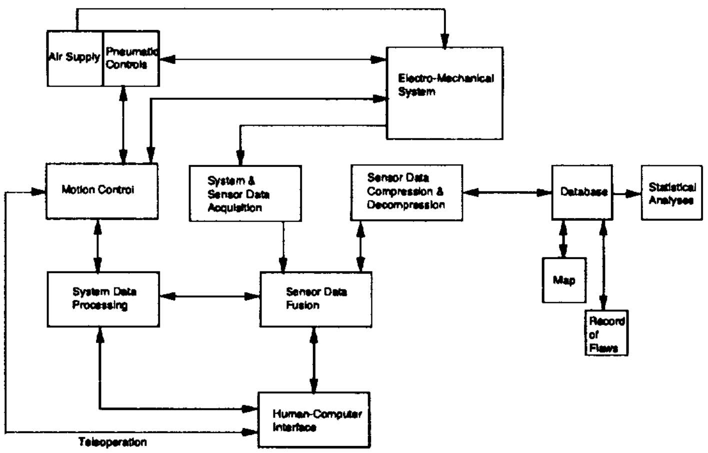


Figure 7: Block diagram of planned complete system.

Special Purpose Actuators

Our initial eddy current sensor deployment demonstration is targeted on part of a mandated inspection on the fuselage of a DC-9 that uses a "reflectance" or "pitch-catch" probe with mechanically independent but electrically coupled transmit and receive coils. This particular inspection has both a surface crack and a subsurface crack component which we address simultaneously by composite dual frequency operation. The reflectance probe geometry is sensitive to integrated conditions over a fairly large patch of skin (a few square millimeters), so it is forgiving of small errors in placement relative to the rivets under examination. Under these circumstances it is adequate to deploy the probe with a simple up-down lifter mechanism and let it self-align with the skin under the influence of a constant-force spring. Another part of the planned demonstration inspection uses a "pencil" probe with a single coil that has a much smaller sensitive area. It must thus be placed and scanned more accurately, e.g., along a path that is tangential to each rivet, which may require closed loop guidance. Another small but necessary part of the demonstration inspection requires moving a pencil probe completely around the circumference of several rivets. The more complex probe paths that this inspection component requires can in principle be achieved by coordinating the motions of bridge-along-spine and bridge-perpendicular-to-spine, but we anticipate that obtaining the necessary mechanical precision and simplicity of control may require adding some nominally

**Messrs Andrew Carnegie and Andrew Mellon both suggest ANDI.

redundant special purpose degrees of freedom, e.g., a rotary mechanism for precisely circumnavigating individual rivets.

Path Control

The path control system addresses mechanical positioning of the eddy current probe and the robot at four distance scales corresponding to the tasks of *alignment*, *guidance*, *navigation*, and *path planning*.

Alignment means the relative position of the eddy current or alternative probe and the rivet or other component under inspection. The inspection protocol is predicated on the assumption that the probe will be moved along a precise short path relative to the part geometry. Signal classification can be done meaningfully only if this path is followed.

Guidance means, for rivet inspection, moving the probe from one rivet to the next and arriving there in correct alignment. For other inspections, e.g., for corrosion somewhere along a skin joint, it means following the required inspection path. In this case it is differs from alignment only in distance scale.

Navigation means coordinating walking and probe guidance so that an inspection that spans multiple robot steps proceeds smoothly and certainly.

Path planning means being able to traverse as rapidly as possible, without inspecting, long distances between areas that require inspection. This is the scale at which collisions with undocumented parts of the airplane (e.g., a non-standard antenna), expected parts in an unexpected state (e.g., an access hatch left open during maintenance), and other maintenance equipment (e.g., a wrench left on a bolt head) are potentially serious problems.

We expect to achieve the necessary position accuracy by dead-reckoning using high mechanical precision motion over short distances between map database landmarks and using machine-vision-based correction at each landmark. The obvious landmarks are the rivets themselves, each of which is in principle individually identifiable in the aircraft design database. The eddy current signals themselves then provide an additional and perhaps finer level of correction: misalignment signatures are recognizable and quantifiable, although some sign ambiguities would have to be resolved by active sensing. Skin joints and skin joint intersections provide additional landmarks. They are particularly appropriate for navigation and path planning, in contrast with the rivets, which are particularly appropriate for alignment and guidance. Skin joints are farther apart than rivets, a disadvantage in terms of dead-reckoning error accumulation, but their existence is more consistent from airplane to airplane (of the same type) since their locations are less likely to be changed by modifications and repairs. The skin joints and skin joint intersections, referenced in terms of the underlying longeron (or stringer) and spar (or body station) identification numbers, are in fact the features in terms of which mandated and recommended inspections are defined.

In principle the map databases are all on-line at the factory for as-designed and as-built, and on-line at the hangar for as-modified and as-repaired. In practice the data are still on paper for all aircraft except the generation now in gestation, e.g., the Boeing 777, and we expect we will have to use ANDI to bootstrap populating its own map and exception database.

Vision System

ANDI will have at least four cameras in the alignment, guidance, navigation, and path planning system. Cameras will also have roles in visual flaw detection, but not until a later phase of the program.

Macro Camera

The first camera will be mounted on the same platform as the eddy current probe, with a macro capability giving it a field-of-view of approximately one rivet. It will be used for fine alignment and for the inspector's visual observation of the appearance of the rivet and the adjacent skin at high magnification. In some inspections that require precision probe alignment the alignment control loop may incorporate the eddy current sensor signal as well. In later phases image understanding will be incorporated for visual flaw detection, and possibly as an adjunct to eddy current or other NDI probes. For example, a particular eddy current probe that has a radially symmetric field geometry may sensitively indicate the presence of a radial crack, but will obviously be blind to its orientation; it may then be useful to use the high magnification camera to find its orientation.

Alignment Cameras

The second and third cameras, each with medium magnification fields of view of about 10 cm x 15 cm, or a line of four to six rivets, will be mounted at the head and tail of the spine. These cameras will be used to locate line segments of rivets. A robust best-fit[18] to the head and tail line segments will guide the eddy current sensor along a scan line. This guidance functionality is required early, so these will be the first cameras installed on ANDI. Guidance is actually required before alignment, because the initial eddy current probe is of the "pitch-catch" or "reflectance" type, which is sufficiently tolerant of misalignment that little alignment (fine adjustment about the guidance line) is likely to be needed. The imagery from these cameras will also be made available to the inspector for opportunistic flaw detection of, for example, lightning holes and small dents. As in the case of the high magnification camera, automation of the flaw detection role for these cameras will come in a later project phase. However, we have already made substantial progress prototyping the computer vision based automation of the alignment function. Several rivet finding algorithms have been tried, including edge detection followed by region growing, gray level variance, and a trained neural network, yielding the general conclusion that even with uncontrolled lighting, low contrast, and interference from specular reflectances, any scale-sensitive operator that has the actual rivet size hard wired into it will succeed. A conventional robust line-fitting algorithm based on minimizing the mean absolute deviation almost always correctly draws the desired line through three, four, or five rivets even for the most ghastly poor images. Early results are discussed and illustrated in the following section.

Zoom Lens Camera

The fourth camera, with an ordinary zoom lens's range of focal lengths and working distances, will be mounted on a motorized pan-tilt head high above ANDI's tail end. In the initial experiments, before general purpose navigation and path planning algorithms are in place, ANDI will be teleoperated between inspection stations. Thus the fourth camera will initially be the inspector's eye on the robot's actual configuration, possible interferences or collisions, sensible paths between inspection stations, and gross visual flaws, e.g., pillowing due to extensive subsurface corrosion. As the program progresses machine vision algorithms will increasingly use this camera for proprioception (visually confirming that the actual robot pose corresponds to the control system's model of the pose), collision avoidance and footfall decisions (new radio antennas, skin patches, or raised head replacement rivets will have to be found, avoided, and entered into the database), long distance path planning between inspection stations, and opportunistic detection of large flaws.

Vision Based Alignment

While there are important exceptions that we will eventually have to address, most of the time rivets line up neatly in evenly spaced rows and columns. ANDI is designed to take maximum advantage of this design rule: what ANDI can do most effortlessly and precisely is to scan an eddy current sensor along a straight line segment parallel to and almost the full length of its spine. The essential alignment problem is thus to align the spine parallel to the line segment under inspection. The approach we are developing is to best-fit visually a short line segment near each end of the spine, best-fit the long line segment to the two short line segments, and scan along the long line segment open loop unless the eddy current data show features that suggest the rivet line wiggles enough that transverse corrections are needed.

On the assumption that if a computer vision algorithm works well with terrible looking images it will probably work better with better looking images, we developed our approach on a sequence of images that we collected with uncontrolled lighting, uncontrolled surroundings (which are obvious in specular reflection), poorly controlled camera standoff from the riveted surface (a test panel with a radius of curvature and other features comparable to a Boeing 737 or DC-9), and a consumer grade 8 mm camcorder camera that we scanned over the test panel by hand. We digitized to 8 bits x 3 colors about 80 frames grabbed from the tape at about 1.5 sec intervals. Each frame was digitized into 480 pixels x 512 lines x 3 colors, then averaged in 8 x 8 blocks into 60 pixel x 64 line x 3 color working images. At the lowered resolution the rivet line segment finding pipeline runs at approximately real-time (1.5 sec/frame) on a workstation. With the camera parameters and resolution we used, rivets are circular blobs that generally fit into a 7 x 7 block. A typical frame, with gray levels computed by averaging the RGB values, is shown in Figure 8.

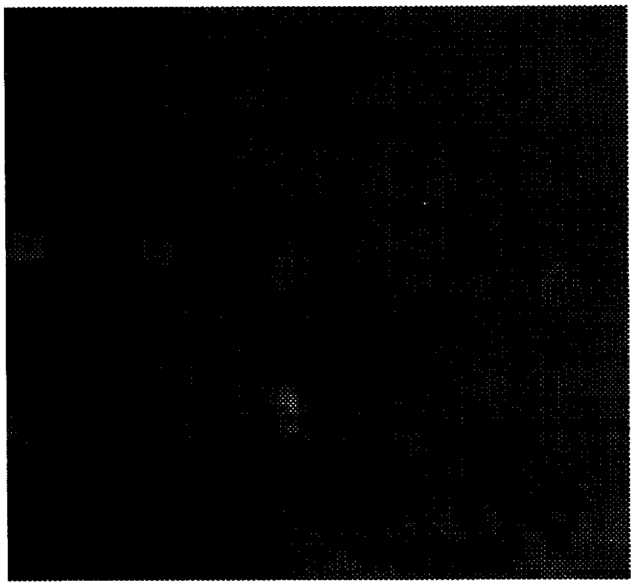


Figure 8: Raw image showing a line of rivets.

Conventional Algorithm

As mentioned in the previous section, finding rivets is easy even when the images are as ugly as this one: any sensible operator with a scale length matched to the rivet size works fine. Under these circumstances a useful strategy is to choose an operator that rarely misses a real rivet even at the price of occasionally finding a false rivet, provided that one or more downstream modules can be tailored to reliably reject false rivets. Finding all real rivets plus some non-rivets we can in fact do with a Canny edge detector[19]. Next we observe that specular reflections, the main potential source of false rivets, look different in each of the three color bands, whereas real breaks in the metal, e.g., rivet edges, have a generally neutral hue. Thus in the second image processing step we reject most of the non-rivets by fusing the three color bands, retaining only those pixels tagged by the edge detector separately in each band. The result is shown in Figure 9.

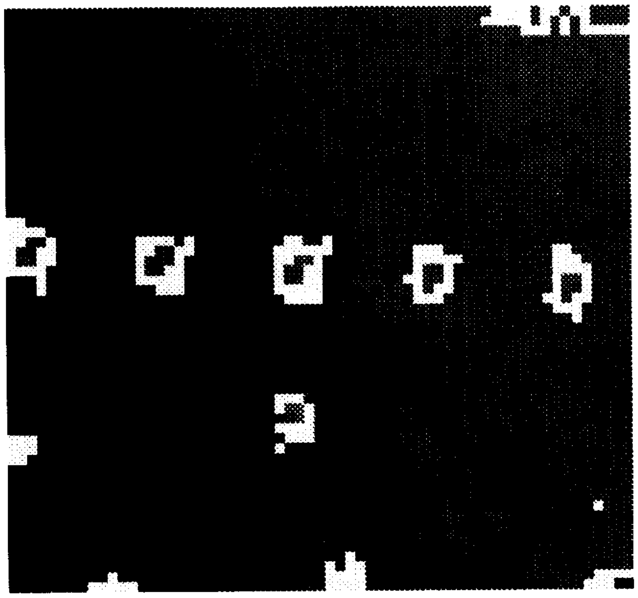


Figure 9: Rivet edge detection by the Canny operator.

Next a region-growing ("grass-fire"[20]) algorithm transforms the perimeters found by the edge detector into blobs filling the areas of the rivet heads. Blobs are rejected if they fail to meet simple geometrical criteria for rivets, e.g., area, aspect ratio, and fill factor within heuristic numerical bounds. The centroids of the surviving blobs are then used as input to a robust (insensitive to outlier) line fitting algorithm[18]. The result is shown in Figure 10.

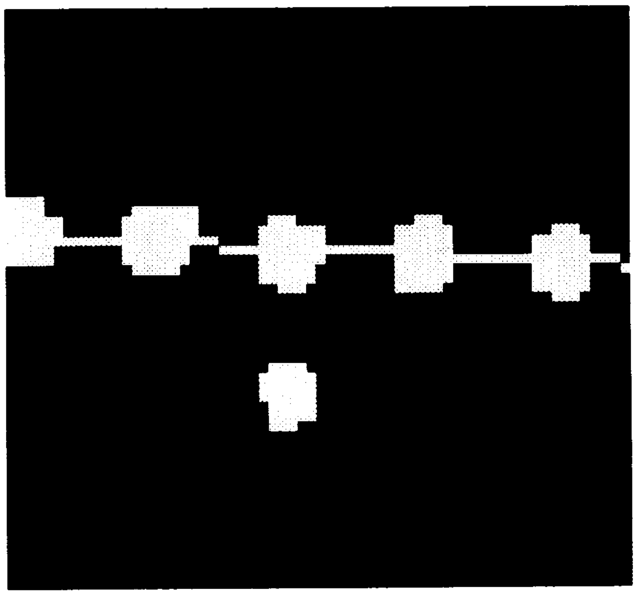


Figure 10: Line fit to the five rivets found.

Neural Network Algorithm

An alternative algorithm was built by training a neural network simulated in software. This method approaches the problem by saying that rather than discovering and finding suitable discrimination parameters and their ranges intuitively, relevant parameters and ranges can be found mechanically by systematically modifying the parameters of a generalized input-output network until it reliably behaves as a rivet/not-rivet classifier when applied to an operator-classified training set representative of the problem; if the training set is adequately representative of the problem domain then the trained network will also be able to classify rivets and not-rivets that were not in the training set.

To implement this method we constructed and trained (using the back-propagation algorithm[21]) a three layer neural net with an input layer consisting of 147 units (a 7 x 7 retina in each of three color bands), five hidden units, and one output unit whose binarized output we interpret as "rivet" and "not-rivet". The network was trained on 40 frames and tested on 40 different frames. Figure 11 shows the output of the trained neural network operator: bright areas are "rivet-like", dark areas are "not-rivet-like". Figure 12 shows the result of thresholding and extracting connected regions of this image, and also the performance of the robust line fitting algorithm on the result.

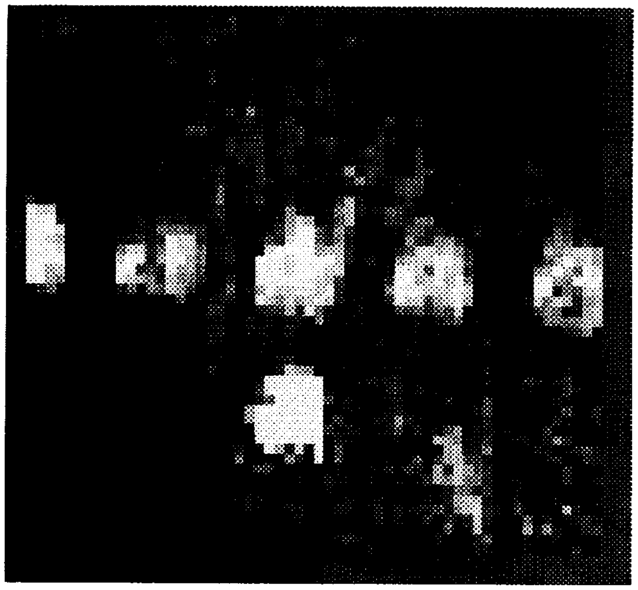


Figure 11: Rivet image found by the neural network.

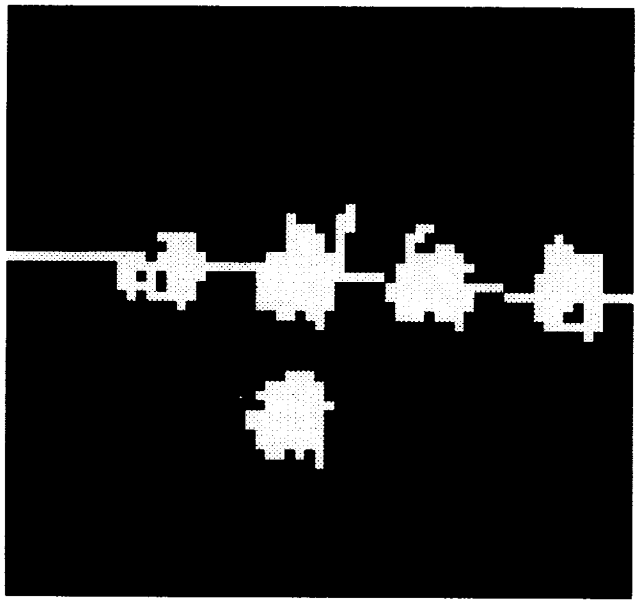


Figure 12: Line fit to the rivets found in the neural network image.

Inspector's Workstation

Figure 4 depicts an inspector's workstation adapted to the environment and culture of aircraft maintenance and inspection. During ANDI's laboratory research and development phases the interim implementation of this workstation is based on two 80486-based PCs. One supports the inspector's mouse-and-menu-based interface, serial communications with the motor controllers, and a general purpose data acquisition and control system with multiple analog-to-digital inputs, digital-to-analog outputs, and digital input and output lines for interacting with various sensors (e.g., suction cup vacuum) and actuators (e.g., solenoid valves controlling pneumatic cylinders). The second PC supports the eddy current probe system and its display. The interim vision system is on an independent proprietary computing platform. It now supports alignment of the robot spine with rivet lines. Its permanent successor will support the additional vision system requirements outlined above.

As development continues the multiple platforms and displays will be rationalized. Our aim is to distribute processing power (which will include providing ANDI with on-board computing power for pose and gait control, etc), and to coalesce the multiple displays by using a powerful windowing system to give the inspector access to controls, signals, images, and data on a single screen. A rudimentary interim database is in place for this function during laboratory tests of vision based alignment, eddy current sensor scanning, and navigation during walking between scanned locations.

Database and Archiving

Aircraft skin inspections are now pass/fail. There is no requirement to record anomalies the reporting threshold. In practice, we are told, airline operators repair all detected flaws, even those below the mandatory and recommended thresholds. Even if this were not the case, pass/fail recording would not necessarily be risky: the thresholds have substantial safety margins, and there are good growth models[22] for predicting how far in the future will repair be necessary. These encouraging practices and circumstances notwithstanding, we nevertheless expect that the predictive capabilities that will follow database archiving and statistical analysis of quantitative inspection results will facilitate maintenance scheduling and potentially increase safety. Thus an on-line distributed database, with an architecture open to access from multiple potential inspection and maintenance locations and tools for trend analysis, improved statistical predictions, and pattern discovery is an integral part of our program. We envision a hierarchical architecture with aircraft type at the top, followed by production series, customer configuration, fleet-wide modifications, and, on an airplane-by-airplane basis, records of individual modifications and repairs. These include individual functional modifications, repair patches, plated regions, regions with oversize replacement rivets, etc. These structural features need to be documented for robot navigation as well as for maintenance.

IV. Visual Inspection

As mentioned earlier, close to 90% of aircraft inspection is visual; our choice of eddy current inspection for the first demonstration of automation to aid aircraft inspectors was driven by the relative simplicity of automating deployment of eddy current probes (and NDI probes in general) in comparison with visual inspection. Unlike NDI, where the goal is usually to detect a flaw whose location and nature is known in advance (from previous experience or from computer modelling), visual inspection has a substantial opportunistic component. The visual inspector's goal is to find not only the anticipated failures, but "everything else" as well: dents, lightning strikes, and other kinds of damage of an unpredictable nature in unpredicatable locations. The open-ended quality of this task makes it an unlikely candidate for a level of automation approaching the level we are planning for NDI.

However discussions with airline management and NDI inspection personnel suggest that an integral visual inspection capability may be perceived as an indispensable component of any economically viable system of automated aids to NDI.

In response to this perception, a mobile end-effector like ANDI does suggest itself as a teleoperable platform from which ground-based visual inspection might efficiently be conducted. If this could be accomplished, it would be valuable for many of the same reasons that ground-based NDI is valuable: reduced set-up time, human-factors issues of inspector performance in a difficult environment, inspector safety, database access, data archiving, etc. The question is whether remote cameras can provide sufficiently high quality (presumably meaning primarily high resolution) imagery to satisfy the notoriously fussy (we are comforted to say) visual inspectors. We recently began a program whose goal is to answer this question. This program combines elements of the FAA-sponsored ANDI project with salient elements of an ARPA-sponsored project in 3D-Stereoscopy Technologies for image and graphics visualization.

One of the costs of human inspection is attributable to the difficulty of safely getting the inspector to the right place on the airplane: it involves erecting scaffolding, providing safety harnesses, etc, all of which can take more time than the inspection per se. ANDI can be placed on an airplane fuselage at human chest level, and directed to move to any area requiring inspection without erecting scaffolding and without endangering the human inspector. Thus even a teleoperated capability, with only the most rudimentary elements of automation (e.g., computer coordination of gait), could permit the inspector rapidly and safely to perform the necessary visual inspections. With appropriately selected cameras and actuators thus could clearly be done at the required variety of points-of-view, magnifications, lighting conditions, etc.

3D-Stereoscopic Vision

We are particularly interested in the prospect of providing the visual inspector with binocular 3D-stereoscopic vision. Stereoscopic perception appears to be important to the visual inspectors who we have observed on the job. We speculate that this may be because of its importance both in perceiving and in rejecting the effects of specular reflection off the mirror-like aircraft skin. Specular reflection appears to be important to inspectors looking for the presence or absence of specific flaws: they often move their heads and lights as the look for an expected tell-tale glint. Specular reflection is particularly apparent in binocular 3D-stereoscopic imagery because the sharply directed reflection appears much brighter in one or the other image, in contrast to the diffuse reflections, whose intensities are evenly balanced in the two images. [For this reason waterfalls and fast running streams, which are notoriously difficult to photograph (and paint) well, look spectacularly realistic in 3D-stereoscopic imagery.] Furthermore, the depth perception provided by 3D-stereoscopic imagery also makes it easy to reject artifacts of the environment that are reflected by the aircraft skin. Without depth perception it is impossible to know (except by high-level knowledge of the context) whether features of the imagery are in the skin or in the environment and seen in reflection. With depth perception, image features that are not in the plane of the skin can be rejected straightforwardly, even automatically.

The components required in a 3D-stereoscopic system are (1) a matched pair of cameras (analogous to the human's two eyes), (2) suitable and suitably controllable lighting, and (3) a display that is capable of directing the image corresponding to the right camera to the operator's right eye and the image corresponding to the left camera to the operator's left eye. There is no ideal way to accomplish (3). Special video taping equipment is also needed if the imagery is to be recorded. A variety of commercially available solutions have pros and cons that we are evaluating in context of the visual inspection application. Available solutions include frame, field, and subfield sequential methods with active shuttering eyewear, field and sequential methods with interline-polarization and passive eyewear, and "virtual reality" approaches using head-mounted displays. We have one subfield sequential and one interline-polarization system operational, so will conduct our initial experiments with these systems.

We are building for these experiments a simple mobile manipulator that will move over an airplane panel test surface according to the inspector's instructions mediated by a computer that will support a suitably high-level interface. A simple mobile manipulator (in contrast to ANDI) will suffice because (unlike ANDI) it will have to operate, for these evaluation experiments, only on a more-or-less horizontal surface.

Acknowledgements

The opportunity to automate aging aircraft inspection was suggested by Stephen A. George. The inspection system concept was developed in collaboration with William M. Kaufman. The design and construction of the ANDI system was done in collaboration with Kaufman, Christopher J. Alberts, Court L. Wolfe, Christopher W. Carroll, and John J. Hudak at Carnegie Mellon Research Institute (CMRI), and Alan D. Guisewite and Ian Davis of the Robotics Institute. The image processing examples are Davis's. Russell Jones and Roy Weatherbee of USAir, and William Keil, formerly of USAir, provided invaluable insight and advice about operating and inspecting commercial aircraft. The design of the 3D-stereoscopic inspection testbed was done primarily by Gregg W. Podnar at the Robotics Institute. The ANDI project is supported by the FAA Aging Aircraft Research Program via FAA Research Grant 93-G-013 and Bureau of Mines Grant G0319014. The 3D-stereoscopic visual inspection project is supported by the Ben Franklin Technology Center of Western Pennsylvania and Aircraft Diagnostics Corporation (ADC) via Grant RR10032 (1993-94). The 3D-Stereoscopic Technologies project is supported by ARPA under the High Definition Systems Program Grant MDA 972-92-J-1010.

References

1. Nelson J. Miller, "FAA Initiatives in Aging Aircraft Reasearch for Assurance of Continued Airworthiness", Tech. report, FAA Technical Center, 1991.

2. *1991 International Conference on Aging Aircraft and Structural Airworthiness*, Federal Aviation Administration (FAA) and NASA, FAA and NASA, Washington, DC, Nov 1991.

3. Richard Johnson, "Aging aircraft and airworthiness", *Aerospace Engineering*, Jul 1992, pp. 23-30.

4. Robert C. McMaster (editor emeritus), Paul McIntire (editor), Michael L. Mester (technical editor), editors, *Electromagnetic testing: eddy current, flux leakage, and microwave nondestructive testing*, American Society for Nondestructive Testing, Columbus OH, Nondestructive testing handbook, Vol. 4, No. xxiii, 1986.

5. Soon-Ju Kang, Nam-Seok Park, and Yong-Rae Kwon, "A Hybrid Expert System for Eddy Current based Inspection of Steam Generator Tubes in Nuclear Power Plant", *Third Symposium on Expert Systems Application to Power Systems (Tokyo-Kobe, JAPAN)*, Korea Atomic Energy Research Institute and Korea Advanced Institute of Science and Technology, Dajeon City and Seoul, Korea, Apr 1991, pp. 144-151.

6. Kil-Yoo Kim, et al., "Eddy Current Signal Recognition Using Neural Network", *Third Symposium on Expert Systems Application to Power Systems (Tokyo-Kobe, Japan)*, Korea Atomic Energy Research Institute, Artificial Intelligence Department, Dajeon-City, KOREA, Apr 1991.

7. C. J. Alberts, W. M. Kaufman, M. W. Siegel, "The Development of a Robotic System to Assist Aircraft Inspectors", *Proceedings of the ATA NonDestructive Testing Forum*, ATA, Airlines Transport Association, Scottsdale, AZ, September 1993, pp. TBD, in press

8. M. W. Siegel, W. M. Kaufman, C. J. Alberts, "Mobile Robots for Difficult Measurements in Difficult Environments: Application to Aging Aircraft Inspection", *IEEE Robotics and Autonomous Systems*, Vol. TBD, No. TBD, TBD 1993, pp. TBD, In press. Invited, based on and slight update of paper[13]

9. M. W. Siegel, "Robotics for *Difficult Measurements in Difficult Environments*: Application to Aging Aircraft Inspection", Robotics Institute Annual Review. (In press)

10. Ian Davis and M. W. Siegel, "Automated NonDestructive Inspector of Aging Aircraft", *Proceedings of the Wuhan meeting*, Huazhong University of Science and Technnolgy, Wuhan, China, Second International Symposium of Measurement Technology and Intelligent Instruments, Wuhan China, October 1993, pp. TBD, in press.

11. Ian Davis, Chris Carroll, John Hudak, Chris Alberts, William Kaufman, M. W. Siegel, "Vision algorithms for guiding the Automated NonDestructive Inspector of aging aircraft skins", *Proceedings of the San Diego Meeting*, SPIE, SPIE, Bellington WA, July 1993, pp. 133-44.

12. W. M. Kaufman, C. J. Alberts, M. W. Siegel, "Automated Inspection of Aircraft", *Proceedings of the Fifth Conference*, Deutsche Gesellschaft fur Luft- und Raumfahrt e.V. (DGLR), International

Conference on Structural Airworthiness of New and Aging Aircraft, Hamburg, Germany, June 1993, pp. TBD, in press.

13. M. W. Siegel, W. M. Kaufman, and C. J. Alberts, "Mobile Robots for Difficult Measurements in Difficult Environments: Application to Aging Aircraft Inspection", *Proceedings of the Pittsburgh Meeting*, C. Thorpe,ed., IAS Conference, International Conference on Intelligent Autonomous Systems: IAS-3, Pittsburgh PA 15213, February 1993, pp. 156-63.

14. C. J. Alberts, W. M. Kaufman, M. W. Siegel, "Automated Inspection of Aircraft", *Aerospace '92, Dallas TX*, SME, SME, June 1992.

15. W. M. Kaufman, M. W. Siegel, C. J. Alberts, "Robot for Automation of Aircraft Skin Inspection", *Proceedings of the International Workshop on Inspection and Evaluation of Aging Aircraft*, Benham Bahr,ed., Federal Aviation Administration (FAA) hosted by Sandia National Laboratories, Albuquerque NM, May 1992, pp. VII 13-18.

16. W. Chun, S. Price, and A. Spiessbach, "Design and Construction of a Quarter Scale Model of the Walking Beam", *Proceedings of the Twentieth Annual Pittsburgh Conference (Pittsburgh, PA)*, Instrument Society of America, Martin Marietta Space Systems Co., Denver, CO, May 1989, pp. 785-788, Modeling and Simulation, Volume 20, Part 2

17. W. Chun, S. Price, and A. Spiessbach, "Terrain interaction with the quarter scale Beam Walker", *Mobile Robots IV (Philadelphia, PA)*, SPIE, Martin Marietta Space Systems Co., Denver, CO, Nov 1989, pp. 98-103.

18. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vatterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge , 1990.

19. J. Canny, "Finding Edges and Lines in Images", Technical Report 720, MIT AI Laboratory, June 1983.

20. R. Duda and P. Hart , *Pattern Classification and Scene Analysis*, Wiley and Sons, 1973.

21. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, The MIT Press, Cambridge MA, 1986.

22. Regis Pelloux, "Tracking Cracks in Aviation Safety - The Facts on Fuselage Fatigue", *MIT Industrial Liason Program*, No. A1191-056, Nov 1991.

-37-23
[illegible]
P. 3

6068

AIAA-94-1224-CP

N94- 30573

Graphical Simulation for Aerospace Manufacturing

Majid Babai
NASA, Marshall Space Center, AL.
Christopher Bien
Deneb Robotics, Inc., Auburn Hills, MI.

Abstract

Simulation software has become a key technological enabler for integrating flexible manufacturing systems and streamlining the overall aerospace manufacturing process. In particular, robot simulation and offline programming software is being credited for reducing down time and labor cost, while boosting quality and significantly increasing productivity.

Graphical Simulation

Simulation technology enables aerospace engineers to capture and evaluate a comprehensive robotic workcell proposal, at a single focal point. This type of functionality, found only in simulation, creates autonomy and facilitates progressive design methods such as concurrent engineering. Used as a concurrent engineering tool real-time simulation of geometrically accurate robots, tools, and peripheral components enables seamless communication between various parties such as design engineers, plant floor engineers, management, vendors, safety engineers, integrators, machine operators, and customers. In addition, many companies are using such tools to create 3-D animated proposals. They are quickly and easily creating simulations of their designs and manufacturing concepts. Contrary to the typical scenario of a customer laboring over a 1200 page proposal comprised of 2-D drawings, charts and descriptions, clients can now interactively preview an accurate simulation of their desired manufacturing process. Ultimately, simulation instills a high level of confidence, understanding, and realistic expectations in the potential customer.

Sophisticated continuous-event simulation technology enables the modeling of geometrically precise and accurately calibrated robots and entire workcells for analyzing every possible scenario. Extremely accurate simulations are possible by incorporating the actual robot attributes such as motion planning, kinematics, dynamics, and I/O logic.

Another advantage that simulation technology has over traditional prototyping methodologies is the ability to save complete libraries of robots, robot accessories, human models, cycle times, and entire workcells on a few megabytes of disc space for future reference or modification. The ability to store and retrieve simulation workcells enables engineers to reuse previously modeled parts, tooling, end effectors, robots, and processes. Simulation encourages "What if...?" experiments and assists engineers in making well informed, confident decisions.

Using these capabilities, NASA has developed a new system for removing the thermal insulation from the space shuttle's Solid Rocket Boosters (SRB) during disassembly at Kennedy Space Center. The thermal insulation is removed from the boosters by high-pressure waterjets, after which the motor segments are separated and sent to the Thiokol manufacturing plant in Utah for refurbishment and reuse. Previously, high pressure stripping nozzles were mounted onto a hand-held gun operated by a technician wearing a bulky protection suit. In order to remove the operator from a hazardous location, and enhance the process, the stripping nozzles were mounted onto a GMF S-420 robot on a mobile carrydeck. The boosters are cylindrical in shape and approximately 149 ft long, with a diameter of 12 ft, consisting of four motor case segments, an aft skirt, and a forward skirt with a frustum containing parachutes.

GMF S-420 robot on mobile carrydeck.

By utilizing IGRIP, robot simulation and off-line programming software from Deneb Robotics, Inc. NASA and USBI (United Space Boosters, Inc.) engineers at the Marshall Space Flight Center created a graphical workcell model of the robotic system and the facility. For modeling complex b-spline surfaces, the engineers first used Intergraph's Engineering Modeling System (I/EMS) to create solid models, which were then directly translated into IGRIP-format polygons for simulation.

NASA was faced with a unique situation. The robot's paths were programmed using the GMF Karel language and could not be tested off-line, due to the expense of creating an accurate full-size model of the boosters. To compound the problem further, the timeline set for disassembly processing (designed to prevent corrosion of the solid rocket motor's steel casing) and expediting SRB turnaround did not allow for extensive development testing to validate the robot paths. Using computer-based solid models and IGRIP simulation software, it was possible to choose the S-420 robot from the robot library and develop accurate stripping paths on an engineering workstation.

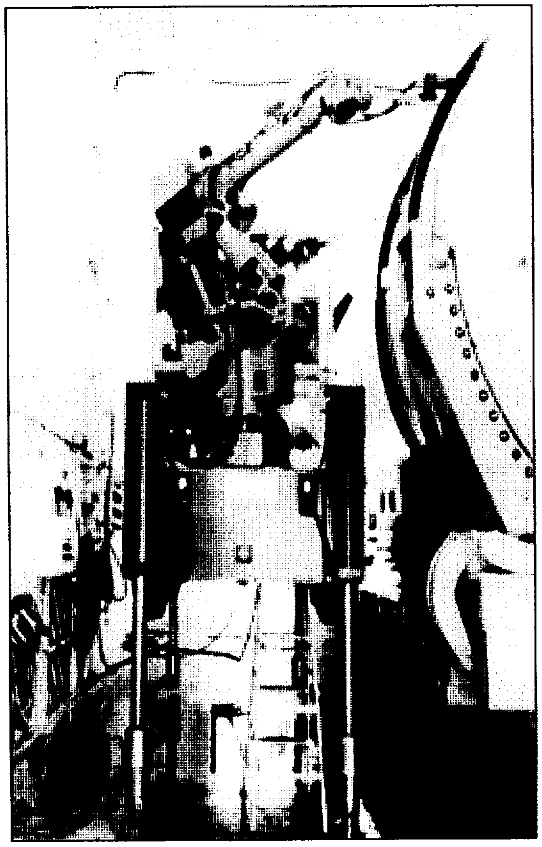
Simulation is instrumental for solving process optimization issues, which include robot motion planning, cycle time prediction, collision detection, and off-line programming.

Robot motion planning was once a speculative risk. During the pre-simulation era, process engineers relied primarily on their years of experience and rules of thumb to estimate critical factors such as the robot speed, joint values, and TCP (tool center point) path. In contrast with past methods, simulation is a powerful decision making tool for determining exact robot motion calculations. By incorporating such elements as inverse kinematics, dynamics, and robot I/O signals into a robot simulation workcell, robot motions can be computed and evaluated with great confidence.

Cycle time is another important aspect of process optimization. Minimizing overall cycle time directly translates into dollars saved. Through real-time simulation, NASA was able to predict run-time lengths of each stripping cycle accurately, before setting up on the plant floor. Here simulation shows its true mettle as an interactive tool for engineering analysis. Questions which once plagued manufacturing engineers such as, "Am I running my machines at the optimal cycle time? Can I speed up my cycle time?" or, "Do I need additional robots?" are resolved through the power of simulation.

Robot collisions are the number-one factor for expensive labor, tooling and downtime costs. The probability of collisions has been dramatically reduced through implementing accurate surface-to-surface collision and near-miss detection and exact minimum-distance calculations found in advanced robot workcell simulation software packages. Collision detection is an ideal engineering tool for identifying the "unexpected" hazards associated with tight tolerances, complex articulated robot motion, and human error.

Collision detection was a serious concern for the NASA team. The boosters contain a pyrotechnic linear shaped charge that is part of the range safety system which is designed to destroy the boosters in the event of a malfunction. To access this linear shaped charge, cork insulation has to be tripped from the system's tunnel covers. Additionally, the robot arm needs to work close to the Thrust Vector Control (TVC) system which power two hydraulic actuators that gimbal the nozzles. Hazardous hypergolic fuel is used to power the hydraulic subsystem; therefore, any leak caused by a collision would endanger personnel. Here, simulation helped to increase confidence and reduce the design cycle for the robot path before actual testing began.



Due to the limited physical workspace, collisions were a serious concern for the NASA team.

Uniquely functional calibration enhancements have made simulation a practical tool for all aspects of automation. User friendly, proven calibration tools are applied to transform "picture perfect" simulation workcells into real-world parameters. By using calibration techniques to mimic the actual world environment, off-line programming has become reality.

The power of simulation and off-line programming is two-fold. First, the robot program is developed and verified in the simulation environment. Second, this program can be translated and downloaded to a specific manufacturer's controller for final touchup. As a result, off-line programming takes a fraction of the time allotted for conventional on-line programming practices.

In the aerospace manufacturing environment, in which product geometry is complex, startup time is limited, model changes are frequent, or the end products are large, off-line programming is the best answer for complex automated production lines and rapid responses to product/process changes. For NASA, simulation capabilities allow engineering responsibilities to be met when configurations on the flight hardware or in the workcell take place.

AIAA-94-1225-CP

6069

N94- 30574

THE AUTOMATED AIRCRAFT REWORK SYSTEM (AARS) A SYSTEM INTEGRATION APPROACH

S43-63

[illegible]

P- 12

Michael J. Benoit
Mercer University
Mercer Engineering Research Center (MERC)
Warner Robins, Georgia

Abstract

The Mercer Engineering Research Center (MERC), under contract to the United States Air Force (USAF) since 1989, has been actively involved in providing the Warner Robins Air Logistics Center (WR-ALC) with a robotic workcell designed to perform rework automated defastening and hole location/transfer operations on F-15 wings. This paper describes the activities required to develop and implement this workcell, known as the Automated Aircraft Rework System (AARS). AARS is scheduled to be completely installed and in operation at WR-ALC by September 1994.

Statement of Problem

The Mercer Engineering Research Center (MERC) was awarded a contract from the United States Air Force (USAF) in January 1989 entitled "Tooling for Fastener Hole Reproduction". The purpose of this task order was the investigation and development of automated tooling concepts to perform fastener hole location on F-15 aircraft wing upper torque box panels, and the subsequent transfer of those locations to replacement skin panels.

The F-15 Eagle Fighter was one of the first modern aircraft to be designed with the aid of computer technology. The majority of the production effort on the F-15 was accomplished through manual methods, following standard aerospace manufacturing practices. Skin panel fastener hole drilling by manual means resulted in unique fastener patterns for each panel, and therefore panels are non-interchangeable among respective aircraft. Unique fastener patterns were not considered a problem until Periodic Depot Maintenance (PDM) rework requirements for the F-15 called for the replacement of wing skin panels.

Due to the non-interchangeability of wing skin panels, replacement skin panels must be supplied blank, without fastener holes, and with excess trim material on the fit edges. The primary reason for this is to allow the rework or repair facility personnel to custom fit the replacement panel to the aircraft. The resulting process is both very labor and skill intensive.

Scope and Methods of Approach

The MERC plan-of-attack for the AARS effort called first for a problem definition effort consisting of the study and evaluation of the F-15 wing PDM rework processes performed at the Warner Robins Air Logistics Center (WR-ALC). The primary goal was to fully understand the manual methodology involved in the hole location/transfer process and to then be able to define the requirements for an automated system.

Basic System Requirements

Based upon the observations of the wing rework processes, the basic system requirements for the AARS were defined, as follows:

-Accuracy/Repeatability: System must be capable of maintaining tight process tolerances over a large work envelope, specifically, to locate hole centers and transfer those locations at less than a 0.005" deviation from the original position.

-Flexibility: System must possess a significant level of artificial intelligence, capable of location and transfer processes on any unique fastener pattern.

-Low Technical Risk: Any system selected must be comprised of proven, reliable technology; simple and easy to maintain.

-Off-Line Programming and Inspection Capabilities: System must possess both of these capabilities in order to verify and validate the processes performed, as well as to integrate the highest quality possible into these processes.

-Ease-of-Operability: Any system chosen must be simple for maintenance personnel to operate, with a minimal amount of training required.

-Safety: Any system chosen must be safe to operate and designed with operator protection in mind.

With this definition of basic system requirements completed, MERC was ready to begin investigations of state-of-the-art robotic technology and vendors qualified to meet the requirements.

Level of Automation

While defining the basic system requirements, the required level of automation had to be resolved. MERC engineers considered varying degrees to which the hole location/transfer process should be automated. Enhanced types of manual tooling aids such as drill blankets were evaluated and eventually rejected because, while the quality of the hole transfer process could be improved, additional tooling setup and takedown time would be required. As well, varying skill levels of the maintenance personnel involved would also effect the level of quality improvement afforded through the use of enhanced types of manual tooling.

Also considered was the adaptation of existing machine tooling. This concept was similarly rejected for a number of reasons, among them that due to the curvature of the wing skin panels, any applicable tooling would have to possess a minimum of five (5) degrees of freedom in order to assure the maintainment of surface normality for enhanced process accuracy. It was therefore determined that while most machining centers and pattern contouring machines possessed the required accuracy, they lacked the necessary flexibility in control and processing required for the task.

It soon became apparent that the only applicable system was one fully automated, or robotic in nature. It was also apparent that fastener hole location and transfer is a process for which any chosen robotic system must possess high repeatability characteristics in order to perform. High repeatability is required in order to properly transfer the fastener hole center locations to the replacement skin panels - necessitating that the chosen robotic system be capable of reliably returning back to the correct hole center location. Due to these considerations, MERC determined that a gantry configuration robot would be ideal for the application, especially since a gantry robot is able to achieve the same level of performance across the entire work envelope.

A System Configuration Merit Analysis was performed by MERC based upon the above discussion, and the results are presented in Table 1 on the following page.

System Technical Requirements

MERC investigated several major manufacturers of gantry robots, as well as machine vision system and end effector tooling vendors. To aid in the final selection of system components, somewhat more comprehensive, system technical requirements were developed:

-Work Envelope Size: A minimum of 18' x 30'.

-Dynamic Referencing/Positioning Capability: Global and Point-to-Point Referencing required.

-System Degrees-of-Freedom (DOF): A minimum of five (5) axes of motion required.

-Controller/Database Capability: Ample data storage, realtime process speed/feedback response, fully downloadable.

-End Effector Tooling: Capability for realtime monitoring of torque, thrust, and dynamic feedback.

Table 1
SYSTEM CONFIGURATION MERIT ANALYSIS

SYSTEM DESCRIPTION	MERIT	COST		SUMMARY
		Material	Labor	
1. Drill guides, plastic templates (transfer media) (Manual)	2	Low ($15,000)	Intensive High skill level required	The Air Force currently uses aluminum transfer templates. They have had poor success at their facility and at McDonnell Douglas' St. Louis facility in using drill guides and plastic templates.
2. Laser scan plotted drill/fastener layout (robotic)	4	Moderate ($250,000)	Intensive High skill level	A laser scan can achieve the required dimensional data necessary to produce a quality plot. Problems include alignment, plot accuracy, plotting size and large human error potential.
3. Touch Probe dimensioning automated drilling (robotic)	7	High ($1,000,000)	Low Reduced skill level	Has advantages in that the required skill level is reduced. Technical problems include probe force, robotic accuracy, interfacing and fixture tooling.
4. Vision/probe hole location automated drilling robotic	8	High ($1,000,000)	Low Reduces Skill level	Has advantages over system 3 in that it can be programmed to adjust to wide ranges of assembly tolerances used in the actual production of the F-15. Same technical problems as system 3.
5. Vision hole location with laster interferometry for volumetric accuracy (robotic)	8.5	High (1,500,000)	Low Reduced skill level	Has advantages over system 4 in that robot repeatability is not as much a requirement because it can be accurately fixed by laser triangulation. Will allow for greater variance in machine.
6. Probe Location Laser Global referencing (robotic)	8.5	High (1,500,000)	Low Reduced skill level	Not as attractive as system 5 because the vision system would most likely be faster than the probe system.
7. Probe/Vision Location with Laser for Volumetric Accuracy (robotic)	9	High (1,700,000)	Low Reduced skill level	The best total system because: 1) flexibility due to vision adjustment 2) probing for exact center location 3) laser can be used for normality 4) volumatic accuracy

System Component Selection

Based upon the technical requirements, the major components of the system were selected as follows:

-PaR Systems XR 225 Gantry Robot: The XR 225 offers excellent accuracy and repeatability performance, with a 225 pound wrist capacity, and the work envelope can be sized to order.

-PaR Systems CIMROC 4000x Robot Controller: The 4000x supervisory controller is based upon an IBM-AT compatible computer using the PC-DOS operating system, and has the advantage of being fully integrated and compatible with the XR 225 robot.

-Adept AGS Machine Vision System: This system affords excellent vision processing capability through efficient handling of variations in lighting, surface finish, and contrast while still providing the image resolution necessary for accurate hole center location. Additionally, the vision system requires minimal effort for integration to the robot.

-EOA Systems CNC Aerodrill: A programmable drilling end effector, offering a full range of performance parameter control, and fully compatible with the robot utilizing the AeroQuick Change adaptor for automated tool pickup and dropoff.

-CENTRO 200 Tactile Offset Sensor: Provides data for referencing by the robot to the part/fixture assembly within the workcell, as well end effector tooling offset data.

-Tooling Fixture(s): Part determinate, and critical to successful automated hole location/transfer operations. The fixture(s) must rigidly support the part to ensure high process accuracy and repeatability.

Table 2 on the following page, illustrates the selected components and vendors as well as their respective system responsibilities.

The individual components selected were all of proven technology, but their integration into a functional automated system for the performance of fastener location and transfer had not previously been accomplished. For this reason, a proof-of-concept effort was performed.

Proof-of-Concept Effort

The primary goal of the proof-of-concept effort was to both demonstrate and validate the capabilities of the AARS to successfully perform automated fastener hole location and transfer operations. Additionally, the feasibility of performing automated defastening with the system was to be demonstrated also.

Specific capabilities to be demonstrated included:

-Proper location referencing and surface contour determination (ie., relative normality of fastener hole locations) of an F-15 outboard wing skin.

-Vision system location (mapping) and storage to the robot controller database of at least two hundred (200) fastener hole locations. Goals for the location/transfer accuracy were less than 0.005" deviation in mapped position, and +/- 1° for surface normality correction.

-Location referencing and surface contour determination of the replacement skin panel, followed by transfer of the mapped fastener hole location via 1/8" pilot holes.

Significant integration effort was required prior to performing the proof-of-concept effort. Probably most critical was that of integrating the vision system to the robot, and development of the vision mapping methodology.

The methodology initially developed for vision mapping operations consisted of generating an AutoCAD drawing of an F-15 upper outboard skin panel utilizing data from McDonnell Douglas assembly drawings. This drawing served to provide initial positioning data to the robot by depicting the nominal locations of the fasteners within 0.125" of the actual physical location. The drawing data was converted via an RS-274D interface to machine control code for interpretation by the CIMROC controller. From this initial positioning data provided to the robot, the actual fastener hole position would fall within the field-of-view (fov) of the vision system camera for mapping. The nominal

Table 2

SYSTEM COMPONENTS	SUGGESTED VENDORS	SYSTEM RESPONSIBILITY
System Integrator	MERC	Responsible for: System Development/Implementation Engineering Tasks, Training of WR-ALC Personnel in System Operation, System Support (Liaison Engineering)
Robot Model XR 225 18' X 30' Gantry	Par Systems	Machine tool platform for hole location/transfer. Main component for tooling integration
System Controller CIMROC 4000	Par Systems	Will control robot during all aspects of both fastener location and transfer operations
End-Effector Tooling CNC Aerodrill	EOA Systems, Inc.	Responsible for robot end of arm tooling
Tactile Offset Sensor CENTRO 200	CENTRO Automation	Will be used to determine surface contour, edge reference, and fixture location data
Machine Vision System	Adept	Responsible for fastener, hole location
Tooling Fixtures	MERC	Will be used to support wing and/or panels during hole location/transfer operations

hole locations provided on the drawings were numbered according to Air Force convention, and this numbering convention was also utilized by the robot controller for databank storage.

The AARS proof-of-concept effort was conducted at the PaR Systems facility in Shoreview, MN utilizing PaR's laboratory setup of an XR 225 robot, CIMROC 4000x controller, EOA Systems CNC Aerodrill, and CENTRO 200 Tactile Offset Sensor (probe). Also utilized was an Adept vision system which was leased for the effort. The proof-of-concept demonstration to the Air Force proved to be very successful, with all goals set for the effort being met or exceeded (see Table 3).

AARS Prototype Development

MERC was awarded the contract for the AARS Prototype Development effort in September of 1991. This contract defined the engineering services required to design, document, and prototype the AARS, and was divided into two phases: a Basic Period (Phase One) and an Option Period (Phase Two).

Phase One Efforts

During Phase One, MERC was tasked to lease or procure the necessary tooling and subsystems to further demonstrate automated fastener hole location and transfer, as well as the additional requirement to demonstrate automated defastening capability. MERC was also tasked to complete a conceptual design of a large wing jig, capable of rigidly fixturing an entire F-15 wing within the robot's work envelope.

The main goal of the Phase One effort was once again to demonstrate the system's capability to perform hole location and transfer, but more importantly, to also perform automated defastening operations. Critical to the successful demonstration of this capability was the performance of vision system.

Vision System Programming

The F-15 upper wing skin panels are tied to the wing substructure with just over 2200 fasteners. The fasteners utilized are primarily comprised of four major types: coin slots, hi-loks, jo-bolts, and taper-

Table 3

GOAL	VALUE	ATTAINED
CAD/Vision Positioning Capability	$\pm$ .125	$\pm$ .125
Hole Location Accuracy	.005 (Global)	$\pm$ .0015
Hole Transfer Accuracy	.005 (Global)	$\pm$.005
Hole Normality	$\pm$ 1 Degree	$\pm$.3 Degree
Bad Hole Determination	Operator Notification	Attained
Vision/Robotic Communication Protocol	Demonstrate Interface	Attained

loks. Coin slot fasteners are threaded, and the preferred removal method is to manually back them out. The other fastener types all require drilling for removal. This difference served as the basis for the required development efforts for a defastening end effector for use with the robot as well as the necessary programming of the vision system.

MERC procured an Adept AGS Machine Vision system during Phase One and performed additional programming of the system to enable it to perform mapping for automated defastening operations. The goal of this additional programming was to provide the vision system with the capability to map the wing skin surface and identify fastener type, size, and location. The programming was accomplished using Adept's Visionware programming environment, through the creation of specific inspection sequences. The basic logic for these sequences was as follows:

1. Locate the object within the field-of-view (fov) and fit an arc to it.
2. Determine the diameter of the fit arc.
3. Determine the center x,y coordinates.
4. Perform a rudimentary inspection to determine if a slot is present.

A serial communication protocol was established between the Adept vision processor and the CIMROC 4000x robot controller for transfer and processing of the vision data. Each fastener hole location on the wing is uniquely numbered, and this numbering convention was maintained for the robot controller database. Since the coin slot fasteners were the only type which required backout, the determination by the vision system as to whether or not a slot was present on the fastener head was the primary criteria for tool selection by the robot controller.

Defastening End Effector Development

MERC was also tasked with the development of a prototype defastening end effector, specifically to perform backout of coin slot fasteners. MERC created a technical specification for the tool, and the decision was made to issue a subcontract to EOA Systems for production of the prototype. EOA Systems was chosen for a number of reasons, among them that the tool would have physical characteristics very similar to that of the Aerodrill, and would therefore be totally compatible with the XR 225 robot wrist. Additionally, the prototype would utilize the same controller as the Aerodrill. The design of the prototype consisted of a stepper motor for slowly rotating the tool tip until the slot was engaged, and a large air pulse motor for backing out the fasteners.

Upon completion of the vision programming and development of the prototype defastening end effector, MERC performed another demonstration of system capability to the Government. This demonstration was again performed at the PaR Systems facility, utilizing their laboratory robot setup.

Capability Validation Demonstration

This demonstration differed from the first not only by the addition of the automated defastening capability, but also in that the system's capability to perform automated vision mapping, defastening, and hole location/transfer operations was conducted on an actual F-15 wing, fixtured within the workcell using a modified Air Force wing transportation dolly. An additional difference was that a Chesapeake Systems Laser Profiler was used in place of the CENTRO 200 Tactile Offset Sensor for normality determination.

The AARS Capability Validation demonstration was conducted over a two day period, concentrating on the outboard skin panel of the F-15 wing. The first day was devoted to demonstrating the system's ability to perform automated defastening operations. This included vision mapping and fastener removal by both backout and drilling.

At the completion of the first day's activities, the outboard skin panel was removed. The second day of the demonstration was devoted to vision mapping of the exposed wing substructure, followed by transfer of the mapped hole locations to a blank skin panel which had been placed back over the substructure. All system capabilities were successfully demonstrated to the Government's satisfaction.

Phase One of the AARS Prototype Development effort was completed with the successful System Capability Validation demonstration, leading to the Air Force's exercising of Option I of the contract for the Phase Two effort in June of 1993.

Phase Two Efforts

With receipt of the AARS Phase Two award, MERC is currently performing a number of required engineering efforts in support of installing the AARS at WR-ALC. These efforts are discussed below.

System Procurement

MERC is procuring an XR 225 robot, CIMROC 4000x controller, and support items (end effector and drill bit racks) from PaR Systems, and a CNC Aerodrill and controller from EOA Systems. Chesapeake Systems discontinued production of their Laser Profiler series, and after an extensive search, a Perceptron Surface Sensor was selected and is also being procured for the system.

Other items required for the system have been identified and are being procured. Among these is a Camera Cable Extender unit from FSR, Inc. It was determined this unit was necessary due to the long length of camera cable which must be installed within the robot (approximately 144 ft.) and the concern that signal loss due to this length would adversely affect the vision system performance.
Also being procured is a custom operator workstation which will house the Adept and Perceptron controllers, as well as the robot, vision, and Aerodrill controller monitors and keyboards.

The system in it's final configuration is depicted in Figures 1 and 2.

Facility Modifications

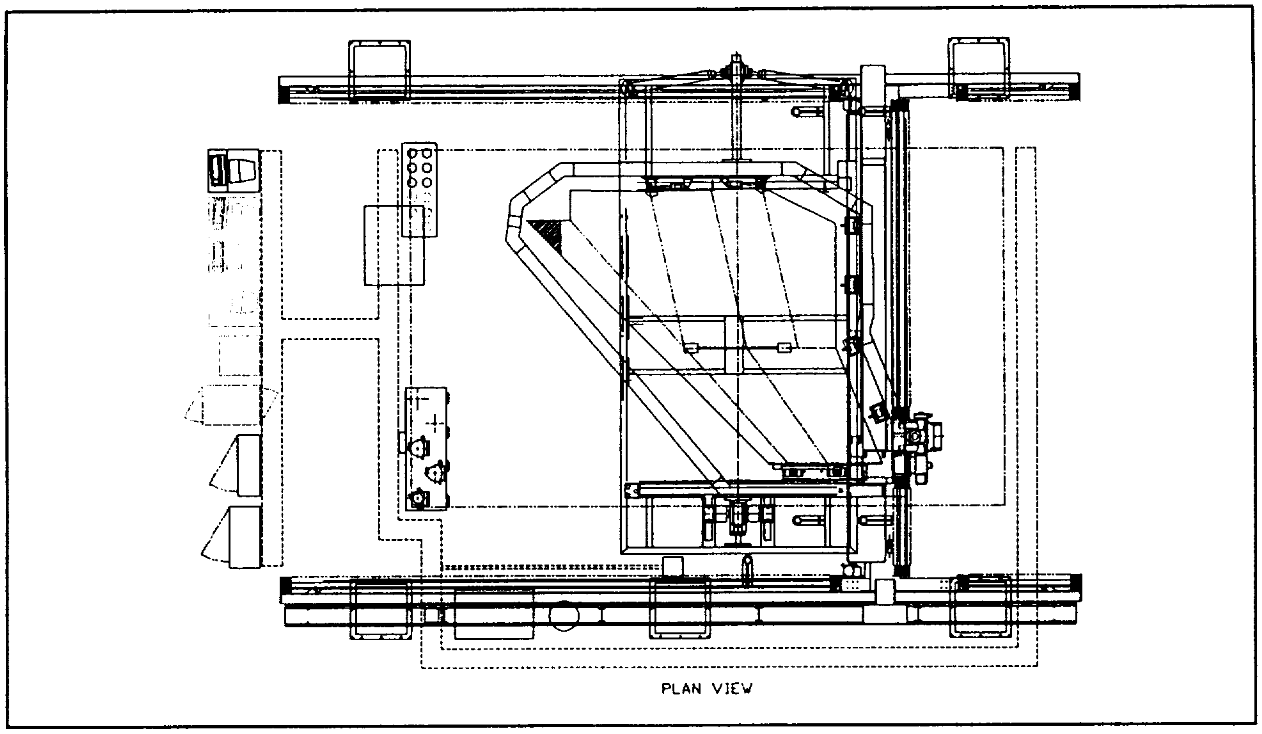
The Air Force has decided that AARS will be installed within Building 140 at WR-ALC, where the majority of rework operations on the F-15 wings are performed. Prior to the actual installation of the system, significant modifications to the facility must first be performed. In order to ensure that the highest level of accuracy and repeatability performance is maintained by the system, the robot must be mounted upon a vibration isolated, floating slab.

The internal work envelope of the robot is 18'x30'. Therefore, the required "footprint" of the system is approximately 30'x40'. At the designated workcell site within Building 140, the "footprint" area will excavated to a minimum depth of five feet. This depth will ensure that the 4 foot thick, 3000 psi concrete slab resting on Unisorb padding is correctly installed. Additionally, utility drops will be provided within 15 feet of the workcell site.

Wing Fixture

MERC completed the conceptual design for a wing fixture during Phase One. Enhancements to this design were identified, and the final design of the wing fixture has now been completed and fabrication efforts initiated.

The fixture design will provide a rigid platform for automated rework operations to be performed by the robot upon the wing. The design allows either a



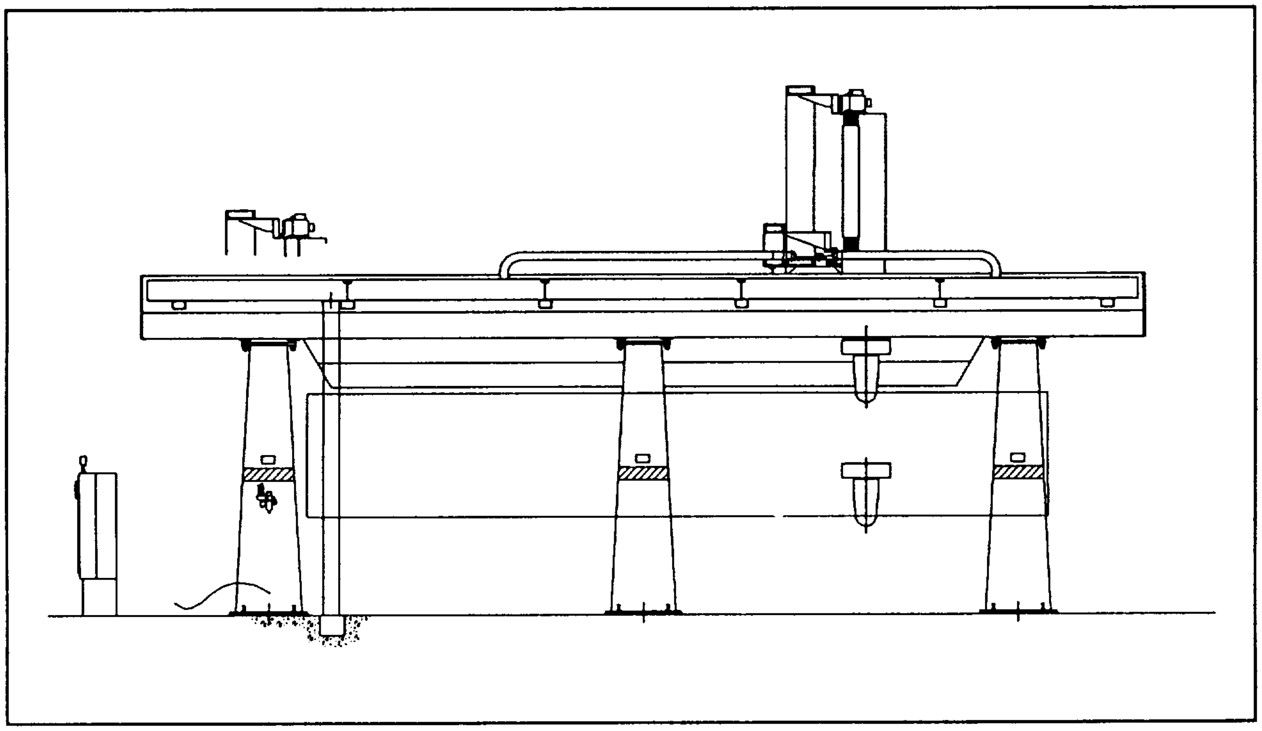


Figure 1 and 2 - AARS System Configuration

right- or a left-hand wing to be fixtured, with access to both upper and lower wing surfaces provided by the rotation capability of the design. The wing fixture design is depicted in Figures 3 and 4.

Subsystem Integration Enhancements

Enhancements to both the vision system and the defastening tool prototype were identified during the Phase One effort, as well as the desired control architecture for the serial communication protocol between robot and vision controllers. The vision system camera and light ring assembly will be mounted in tandem with the Perceptron Surface Sensor to an EOA Quickchange tool plate for compatibility with the robot wrist. This tool plate will be "pinned" by EOA so that all power and communication leads can routed directly through the wrist, enabling automated pickup and setdown by the robot.

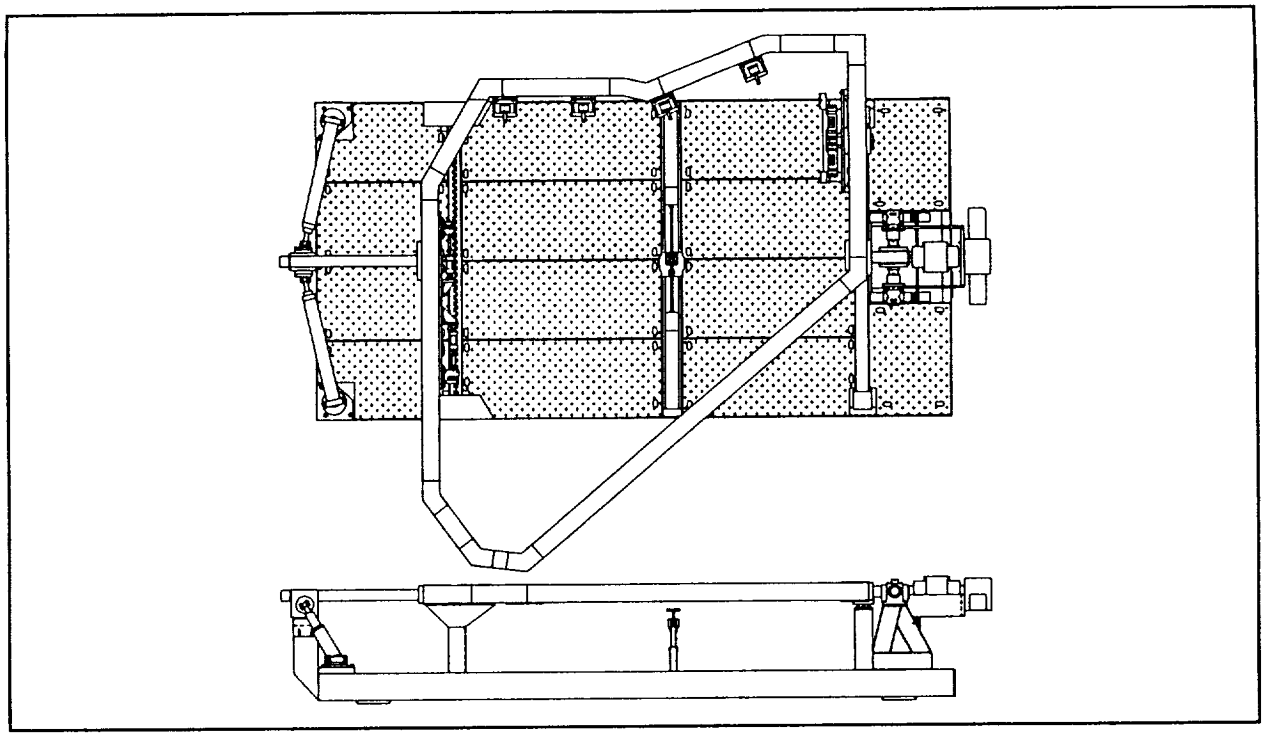
Vision system programming will also be enhanced, by customizing the inspection sequences created in Visionware with Adept's V+ line code. This additional programming will provide the vision system with a minimum level of artificial intelligence, enabling it to more optimally perform mapping operations through the automatic varying of parameters such as gray scale and binary thresholds. The system will also be able to vary it's primary search areas within the focused field-of-view automatically in order to compensate for different fastener head and hole sizes.

The level of serial communication protocol between the CIMROC and Adept controllers is being enhanced to further define and implement a more comprehensive level of post processing capability to include error handling and data validity checking. For example, the vision system and/or laser sensor will return to the robot controller process data and whether or not the data is valid. The robot controller will accept a list of parameters or data fields which will be stored within the record for archiving purposes. The validity of the data will be based on a single binary bit (0 or 1)
sent by the peripheral equipment to the CIMROC. The CIMROC will mark any data records in error and print the record number (per Air Force numbering convention) to hard copy if the validity of the data is NO. The system operator will then scan this hardcopy error list, decide the best course of action, interact directly with the peripheral equipment to alleviate the problem (ie. reprocess a vision image), rehabilitate recoverable data records, and mark those records which are irrecoverable.

Modifications are also being performed to the defastening tool prototype to incorporate enhancements identified during the Phase One effort. These enhancements primarily involve enabling the tool to utilize a two-step removal methodology for backing out the coin slot fasteners. During Phase One, some fasteners were stripped, or had the heads rounded off due to the inability to vary pressure to the large air pulse motor utilized. Tool modifications will include swapping out the stepper motor for a more powerful 2.5 hp spindle motor, and mounting a solenoid on, or very near the end effector to precisely monitor and vary pressure to the air pulse motor. Lessons learned during Phase One will also be employed so that the spindle motor will not only serve to locate the tool tip into the slot, but also as the primary backout tool. In the event that larger, or stubborn fasteners cannot be "broken free" with the spindle motor, the air pulse motor will used for very short intervals, or bursts, to breakout the fasteners.

Process Development Support

MERC is actively engaged in assisting WR-ALC personnel with preparations for the installation and optimal utilization of the AARS workcell. This support includes conducting working sessions with WR-ALC engineers and maintenance personnel which have served to help develop initial implementation procedures for the workcell. These procedures define use of the robot with both new and existing rework tooling and resources, as well as the recommended initial work volume to be scheduled using the workcell. The working sessions have also aided in the selection of qualified personnel to be trained as system operators. Additional support is being provided through recommended revisions to the F-15 wing rework Work Force Order (WFO) documentation, which will address issues including the effect that use of the workcell will have on rework flow time per wing and optimal process insertion recommendations.



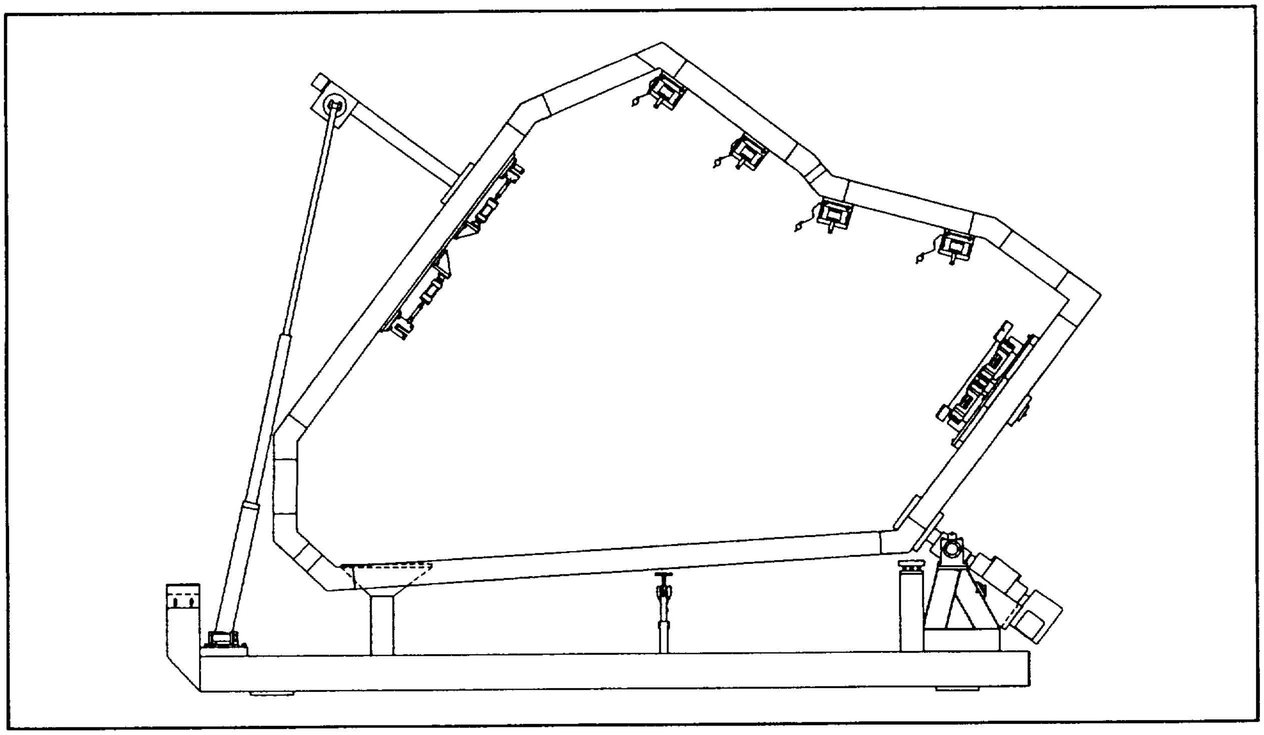


Figure 3 and 4 - AARS Wing Fixture Design

System Installation and Checkout

Workcell installation at WR-ALC will take place in June 1994. The system will be erected by MERC and PaR Systems personnel, and will undergo a procedure known as Mechanical Error Correction (MEC), during which a laser system will used to precisely align and level the robot for optimal accuracy and repeatability performance. The functionality of the robot and all peripheral equipment will then be completely verified following comprehensive qualification test procedures. Additionally, the wing fixture will be loaded with an actual F-15 wing and vision/laser mapping, defastening, and hole location/transfer trials will be conducted. At the completion of the performance trials, MERC will follow the Government approved Acceptance Test procedures and will conduct a formal Acceptance Runoff of the system for WR-ALC officials.

It is anticipated that MERC will spend the remainder of the program schedule (approximately two months) after system acceptance onsite, assisting WR-ALC maintenance personnel in familiarization with the system and its optimal use and benefit to the F-15 wing rework effort.

It is also anticipated that the AARS engineering prototype and supporting data developed during the Phase Two effort will provide sufficient information to the Air Force to support a decision to procure production configurations of this equipment.

Summary of Important Conclusions

The Air Force specified that the AARS have the capability to transfer hole locations to new structure within 0.005" of existing mate-with holes, and to maintain specified hole diameter tolerances (+0.0022"). Results achieved during the two laboratory demonstration efforts indicated a vision mapping location accuracy of +\-0.0015", and an average transfer accuracy of +\-0.0029". It was determined after remapping with the vision system that hole diameter tolerances were maintained to within +\-0.001". These results are even more encouraging when taking into account that the laboratory robot setup used is an older system lacking later generation refinements, has not undergone the MEC procedure in several years, and also is not mounted on a vibration-isolated slab.

The results of the automated defastening trials performed with the AARS indicated a 100% success rate, with all fasteners being removed through either backout or drilloff methodologies. Additionally, absolutely no damage to skin panels or substructure was incurred as a result of the defastening process. This is very important, in that a significant number of wing skin panels requiring replacement have resulted from organic rework damage, or specifically, damage incurred during manual defastening operations.

Implementation of the AARS workcell into the F-15 wing rework effort at WR-ALC will result in both significant enhancements to rework process quality and a marked reduction in required manhours. (see Figures 5 and 6) Additionally, the AARS has been designed with flexibility and future expandability in mind, and possible future applications already identified include the rework of additional F-15 and other aircraft components, automated NDI operations, and fuel foam removal operations.

Lastly, it is projected that full amortization of the total system investment costs will be realized within the first full year of operation.

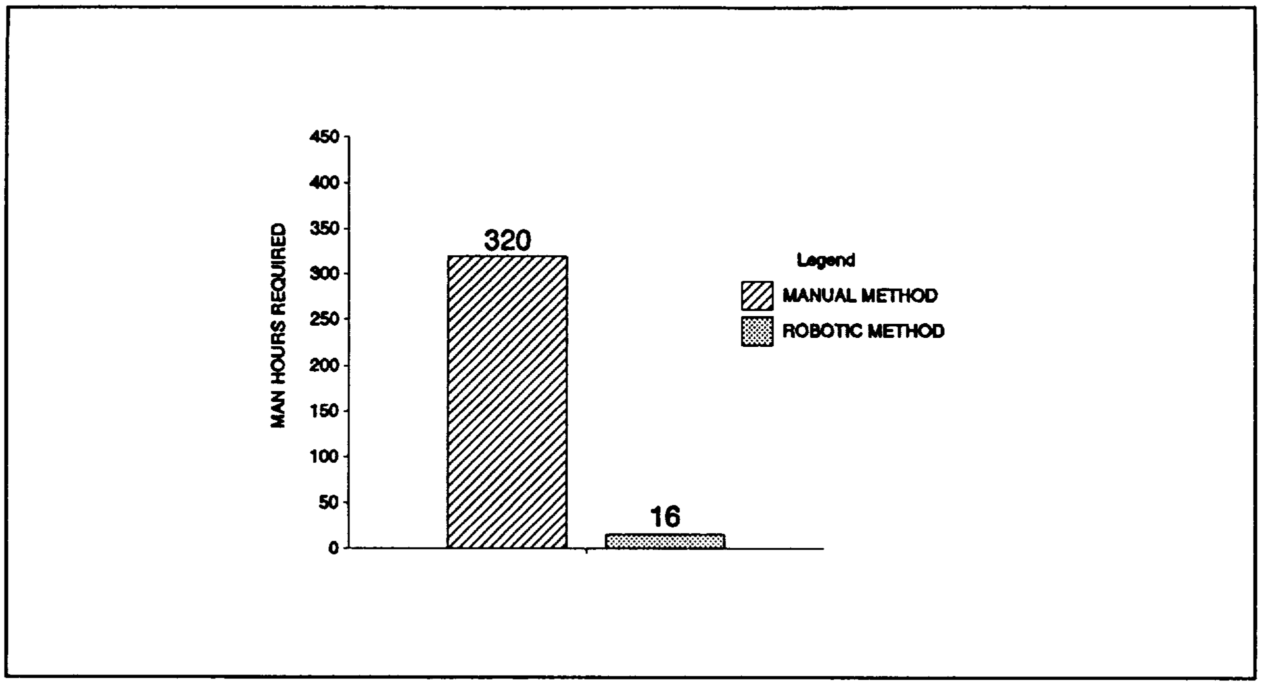


Figure 5 - Required Manhours Comparison
Automated vs Manual Hole Location/Transfer

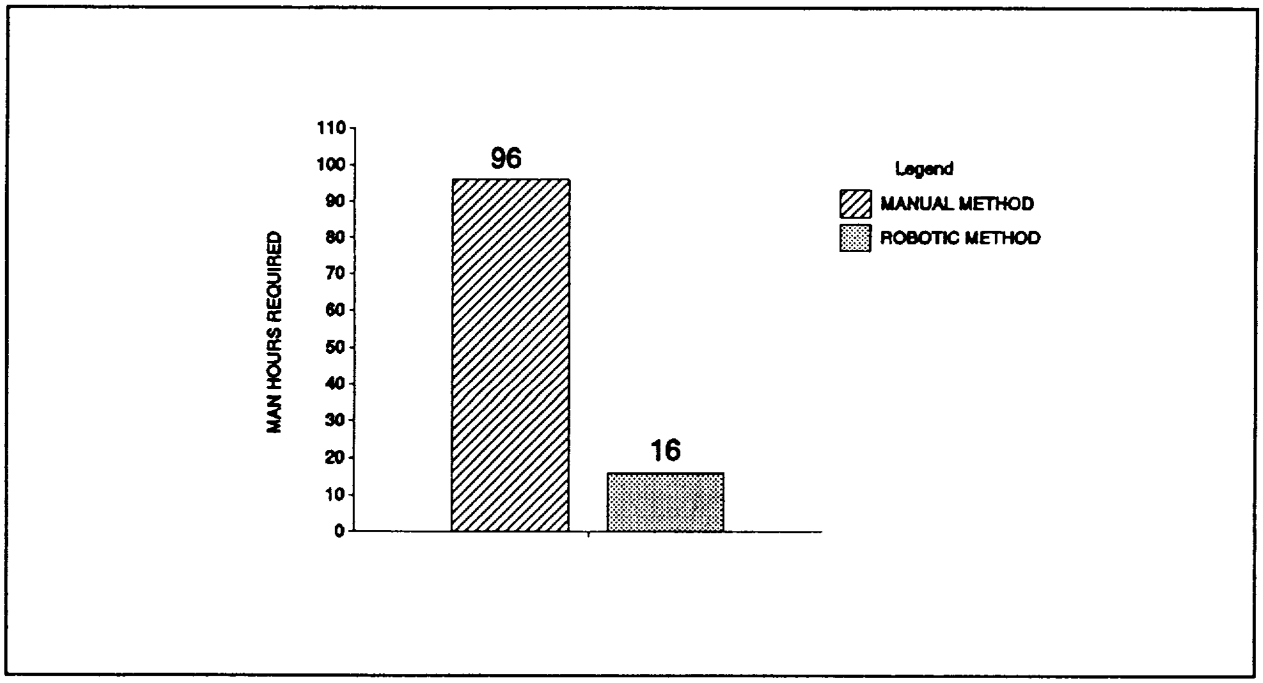


Figure 6 - Required Manhour Comparison
Automated vs Manual Wing Skin Defastening

1070

AIAA-94-1226-CP

N94-30575

AUTOMATED INSPECTION OF TURBINE BLADES: CHALLENGES AND OPPORTUNITIES

72

p. 6

Manish Mehta, Joseph C. Marron, Robert E. Sampson and George M. Peace
Environmental Research Institute of Michigan
Ann Arbor, Michigan

Abstract

Current inspection methods for complex shapes and contours exemplified by aircraft engine turbine blades are expensive, time-consuming and labor intensive. The logistics support of new manufacturing paradigms such as integrated product-process development (IPPD) for current and future engine technology development necessitates high speed, automated inspection of forged and cast jet engine blades, combined with a capability of retaining and retrieving metrology data for process improvements upstream (designer-level) and downstream (end-user facilities) at commercial and military installations. The paper presents the opportunities emerging from a feasibility study conducted using 3-D Holographic Laser Radar in blade inspection. Requisite developments in computing technologies for systems integration of blade inspection in production are also discussed.

1. Introduction

The factory automation of compressor and turbine blade manufacturing for aircraft jet engine power plants is a process that involves the integration of a variety of subsystems into the complete manufacturing and refurbishment cycle. Once processing parameters, envelopes for variation and control strategy are reliably established, perhaps the most important area of challenge lies in the rapid sensing and acquisition of dimensional and surface quality attributes of blades that validate the design, processing and control strategy.

The success of new technology development programs such as Integral High Performance Turbine Engine Technology (IHPTET) and the economic viability and competitiveness of current military and commercial engine blade production rests heavily on the ability to develop and implement cost-effective automation solutions that are highly reliable, and lead to consistent quality components at minimum cost. The ability to rapidly achieve automated blade inspection opens up a world of possibilities that translate to our ability to retain vital information on individual blade attributes for utilization both, upstream (in the design and processing iterations) as well as downstream (in process control, field use and maintenance). An integrated product-process development (IPPD) approach like the one envisioned in the IHPTET program has the potential to lead to a 50% or greater reduction in manufacturing costs[1]. In such advanced engine programs, the criticality of consistently meeting design parameters and tolerances in manufacturing, tracking changes during maintenance and overhaul, and validation of manufacturing process models cannot be overemphasized. An automated blade inspection system on the shop-floor when integrated with a design and processing database through a Computer-Aided Design (CAD) 'reverse-modeling' capability, could be responsible for accomplishing a large portion of that cost reduction through its support of IPPD and Knowledge-Integrated Design Systems of aeroengine components.

1.1 Present Blade Measurement Technology:

In current forged compressor blade and investment-cast turbine blade (Figure 1) manufacturing environments, all metrology and in-process inspections are performed off-line by large pools of human labor. One recent plant study performed by ERIM determined that as much as half of the plant labor were involved in manual inspection and rework of blades. Dimensional inspection is limited to use of hard go/no-go guillotine gages that contact a forged or cast blade at several consistent locations, at each of which the 'fit' is probed by the inspector, and rework locations marked- a process that takes even the expert up to three minutes per blade.

A human inspector is highly flexible, and has relatively fast recognition and decision-making capability. However, the task of manual turbine blade inspection is particularly challenging because of the inspector's high susceptibility to a lack of concentration over several hours, which results in a relatively poor average performance of the person. Furthermore, different inspectors have been known to arrive at different dimensional inspection results, producing variable final batch outputs that are unpredictable and inhibit process and resource optimization. In such an environment, statistical process control is especially difficult to implement. The critical nature of the application to commercial and military aircraft propulsion requires that parts be inspected 100-percent. Present coordinate measurement machines (CMMs) impose an inherent and obvious limitation on the inspection throughput achievable, except for process certification and qualification. By 100-percent inspection of blades, an automated sensor system can be used to control even a slow decay in quality that cannot be found easily by off-line statistical checks.

Following represent typical tolerances for key compressor blade sections used in advanced engine programs:

Twist	+/- 30'
Platform	+/- 0.18 mm (0.007 in)
Chord	+/- 0.25 mm (0.01 in)
Thicknesses	+/- 0.09mm(0.0035 in)
LE Profile	+/- 0.05 mm (0.002 in)
Bow	+/- 0.05 mm (0.002 in)

1.2 The Economic Case for Automation:

The following represents a typical production inspection scenario at a leading aircraft engine compressor blade forging plant (also applicable to a typical turbine blade investment casting facility):

- The plant possesses tooling capacity for approximately 1600 different blade designs, of which about 400 designs are processed in batches at any given time, with a weekly production rate reaching between 60,000 and 90,000 parts.
- Compressor blade forgings of titanium may be up to 0.45 m (18 in) in length and 0.2 m (8 in) wide, with the majority (80%) fitting an envelope size of 0.20 m (8 in) length and 0.06 m (2.5 in) width. A typical compressor blade assembly is shown in Figure 2.
- Each blade is inspected twice by human inspectors using two sets of gages (one spare for use when the other is being re-certified or recalibrated). The plant's metrology equipment investment per blade design is slightly over $400,000.
- Work areas are often filled with dust, smoke, noise, vibrations, or heat that can additionally affect measurement system design, and consequently, cost and system life.
- The cost of quality (i.e., scrappage) ranges from $500-$1000 per blade.

Despite operating under strict quality control guidelines, high quality requirements have frequently been met at such plants by scrapping large quantities of blades[2]. Thus, there is potential for realizing significant cost savings by interception of nonconforming product earlier in the manufacturing cycle using automated inspection.

1.3 Sensor Selection Criteria:

Following minimum criteria are considered important in sensor selection for automated blade inspection:

- Inspection speed or cycle time
- Part throughput
- Sensor parameters such as resolution, repeatability, accuracy, range ambiguity, etc.
- Robustness to plant-floor conditions (temperature, vibrations, dust, operator handling)
- Costs and life (acquisition, installation, training, maintenance)
- Flexibility in measurement of blade designs

2.0 Non-Contact Blade Inspection Systems

2.1 State-of-the-Art:

A recent ERIM study[3] determined that many 3-D measurement systems are available commercially for industrial and metrological inspection, most based on optical sensing techniques such as: (1) intensity modulation laser scanners, (2) stereo, (3) structured light sensors, (4) Moire sensors, (5) holographic sensors, and (6) interferometric sensors. Most commercially available 3-D sensors are based on the triangulation principle, including structured light, stereo and Moire. 3-D sensors employing laser modulation and interferometry have also been commercialized, but to a lesser extent.

An ERIM-developed intensity or amplitude modulated laser scanner, supported by the CYTO-HSS pipeline cellular-array image processor, has been demonstrated in forging operations for missile nose cone components[4,5]. The scanning system circumvents the complex computational and conceptual difficulties associated with 3-D reconstruction of the imaged component from a limited number of camera views. One major shortcoming of laser scanning technology identified was the high level (and hence, expense) of maintenance required to keep the highly polished mirror and optical surfaces clean in the forging environment. In stereo imaging, triangulation is the most commonly used method for 3-D sensing. However, stereo techniques require some common reference point for the two camera views to be correlated together- if the surface being inspected is a smooth, continuous curve (as in airfoils and blades), such a point may not exist. Even when located, the accuracy in determining the single point is limited by the resolution of the camera, and does not provide information about other points on the surface in question[6]. Compared with laser scanning, Moire techniques[7] are known to provide better dynamic-depth range and repeatability, while using an eye-safe white light. They are best suited for inspection of objects that have limited depth such as the body panels of automobiles. Besl[8] has discussed details of evaluation criteria and applications of several optical range imaging sensors. Holography-based sensors, capable of meeting the challenges of precision industrial inspection. One such system is 3-D Holographic Laser Radar (HLR), which is discussed in greater detail in the following section. While structured light active triangulation 3-D sensing is a very flexible technique, there is a risk of shadowing and obscuration occurring, particularly for objects with step and pole-like features as are often encountered in inspection in the vicinity of the airfoil platform[3]. A new 3-D sensing product from Perceptron, the LASAR[R,9], couples laser radar technology with a precision scanning mechanism and control/display software. The result is a system with a programmable field of view that can simultaneously capture both a 2-D image, based on a standard reflectance phenomena, and a 3-D image based on range data. Olympus[10] has recently introduced a limited capability for performing off-line measurement of manufacturing defects on blade surfaces by using memory-stored wire-frame models that are 'superimposed' on the captured image of a blade.

2.2 Holographic Laser Radar:

ERIM has recently experimentally demonstrated 3-D Holographic Laser Radar[11] (HLR) a technology derived from synthetic aperture radar[12], that shows promise in meeting some of the most stringent accuracy, resolution and performance requirements of a blade production environment. The equipment setup and image recovery process are illustrated in Figures 3 and 4. HLR uses a frequency-tunable laser source (such as an argon dye laser) and holographic recording methods to recover digital 3-D representations of imaged objects. Essentially, the sensor measures the 3-D complex object spectrum by gathering data at different spatial frequencies (up to 64 have been demonstrated at ERIM). At a given wavelength, a 2-D (X-Y) slice of the object spectrum is measured using a detector array that is placed perpendicular to the line of sight towards the object origin. Changing the measurement wavelength, a different slice of the object spectrum is acquired which is off-set in the ranging direction. Thus, by sweeping through a set of wavelengths separated by a constant amount over a total allocated bandwidth, a cube-like volume of Fourier data are gathered. Using the inverse Fourier transformation, a 3-D image of the object is formed.

2.2.1 HLR Sensor Attributes for Blade Inspection:

A preliminary technology assessment performed by a concurrent engineering group at ERIM[3] has revealed that the HLR technology effectively addresses several challenges in blade inspection through the following features of the sensor:

(1) The angle-angle measurement accuracy of the sensor is decoupled from the ranging accuracy, which is important when large parts are to be measured to a high degree of accuracy. The sample permits different sampling densities in the angle-angle and range directions, providing both, dimensional and surface finish information.

(2) An absence of imaging optics as well as scanning mechanisms (i.e., no moving parts) further enhances the desirability of HLR in production inspection of blades. The HLR samples the object light field array directly with a CCD detector array, and the positioning of the detector pixels in a detector array can be very accurate.

(3) If a collimated beam or plane wave is used as the reference, the image formed by 3-D Fourier transformation is in a rectilinear format. This feature is unlike most other types of 3-D sensors discussed above which produce images in polar format or non-linear grid, requiring extensive coordinate transformations for conversion into a usable geometric object representation.

(4) When a point source is used as the reference, its location provides a fixed reference point from which all 3-D measurements are made. This is critical for metrology applications such as parts-to-CAD models, where images from different views must be fused together.

An ERIM HLR sensor was used in a feasibility exploration for initial dense surface profile image collection, as well as to characterize the requirements of HLR technology applied to turbine blade inspection. The 3-D images of both, investment cast and forged blades were acquired by varying the following parameters to determine optimal operating conditions: range resolution, range ambiguity interval, and angular resolution.

2.2.2 Sensor Design and Economic Issues:

Forged titanium compressor blade leading edges are a special subset of dimensional measurements inspection because of the difficulty posed by specular surfaces that challenge other competing measurement technologies (Moire interferometry, laser triangulation, etc). Any solution for this problem can likely be applied to castings, wax models and die cavities. HLR performance data for a variety of surface finish characteristics resulting from investment cast, forged-shot peened, and machined blades is presently being evaluated to determine optimal equipment operation bounds, so as to generate the sensitivity and dynamic range requirements for precision metrology applications. Other design issues that need to addressed in inspection, include the scalability for measuring larger objects, and sensor stability in the production scenario. As a coherent sensor, mechanical stability within the HLR and between the sensors and test object must be kept to a small fraction of an optical wavelength. Phase coherence must be maintained not only over the integration time of each spectral measurement, but over the entire imaging sequence through all the measurement wavelengths. This requirement necessitates stringent vibration isolation and the use of an enclosure to minimize air turbulence.

The HLR sensor can be effectively designed to address the metrology requirements of at least 80% of forged and cast blades, which fall within the 0.2 x 0.06 x 0.025 m (i.e., 8 x 2.5 x 1 inch) size envelope. To meet future desired measurement accuracies of up to 0.025 mm (0.001 in) on the airfoil and blade dovetail platform, would require an extremely large number of measurement points over the objects. The slopes of the blades, however, vary gradually, which may allow fairly coarse spatial sampling in the X and Y directions. Moire sensors cannot take advantage of this feature because of the tight coupling between the angle and range measurements.

An economic analysis of automated blade measurement cell using HLR is presently being performed, considering the projected life cycles of key system components. A production sensor incorporating the 3-D HLR technology is projected to cost between $50,000 - $100,000. Additional costs quantifying the impact of the sensor technology on the measurement workforce, production line balance, plant layout and process flow changes in a forging or casting facility, as well as other in-plant implementation and culture issues, are also being addressed.

3.0 Automated HLR Blade Inspection Workstation

3.1 System Requirements:

The aim of an automated metrology and inspection system for turbine blades should be to integrate the components into one complete system. The integrated system should have the computer processing, data storage, and retrieval power needed to inspect, assemble and control all tasks without significant human intervention. An automated blade inspection workstation (Figure 5) incorporating 3-D HLR, as envisioned by us, would consist of four major building blocks:

- Parts handling system that positions blades for inspection and final sorting,
- Laser source system to generate radiation,
- Detector array that converts reflected radiation into a signal or data for host computer processing, and
- Host computer with operator interface and electronic linkage to blade design database,
- An image processor for analysis and decision making.

3.2 Part Fixturing and Orientation:

The HLR sensor provides output data in terms of 3-D rectilinear coordinates for each sample point, as opposed to multiple transformations from polar coordinates that are required for laser scanners. By generating dimensional measurement data on the front/back surface area of the airfoil, it is possible to provide the equivalent measurement capability of not only the contact-type guillotine gage, but also the combined (and potentially enhanced) measuring capability of the blade thicknesses, bow, twist angles, and orientations relative to platform.

3.3 Software Requirements:

A formidable task in developing a systems solution for automated inspection of turbine blades lies in addressing software requirements for registering data from multiple images, extracting and mapping metrics into the gaging measurements and features of interest to the user. This task could also involve interaction with commercial surface mapping and visualization software, in order to determine optimal data density for accurate surface reconstruction and subsequent comparison with CAD and surface design databases. The key would be to determine the approximate X-Y sample spacing required to extract desired surface contour measurements.

3.4 High-Speed Image Processing and Sensor Fusion:

Automated surface and 3-dimensional inspection of turbine blades in a production environment requires real-time-mode image processing, as the number of operations required to achieve defect recognition and metrological interpretation of sensed data tend to be enormous. For example, a high-resolution 3-D HLR image of an entire turbine blade can generate approximately 4 Mbyte/sq in. for a 0.0005 in sample spacing. Decreasing the spatial frequency of the sample points reduces the data size, however, at the expense of accuracy in locating edges. It would be a waste of time to first store dense blade image data into an image memory, and then process it later by accessing the image memory. Thus, it is a requisite in automated blade manufacturing/inspection operations that a fast method be deployed with adequate hardware support such that the completion of image acquisition implies the completion of the image processing (or at least preprocessing). Current trends indicate that a 200-MIPS microprocessor and a 1 gigabit memory chip are soon to become available to meet such image processing challenges, well before the end of the century if lithography limits and other difficulties in each generation are overcome smoothly[13].

4.0 Conclusion

The study has demonstrated that the key technologies and system components for realizing fully automated inspection capability for accomplishing dimensional measurement and defect location in compressor and turbine blades already exist. A systems design, engineering and integration approach is required to evaluate proposed inspection methods in greater detail and to develop alternative methods to satisfy manufacturing and metrology support requirements of commercial and military users. A fully automated Holographic Laser Radar inspection system could dramatically outperform the current manual inspection process by improving the consistency of the inspection process and raising the quality of the blades in service.

5.0 Acknowledgement

The authors gratefully acknowledge the support provided for this study by the Environmental Research Institute of Michigan's Broad Initiative in Metrology Program, and the NASA CCDS for Space Automation and Robotics(SpARC).

6.0 References

1. Anonymous, "Manufacturing 2005: A Strategy for a Strong, Responsive Air Force Industrial Base, Vol.II-Sector Assessments", US Air Force Materiel Command, 1993.
2. Anonymous, "Data Acquisition Ups Blade Quality", Quality in Manufacturing, November-December 1993, pp.34.
3. Tai, A., "Three-Dimensional Sensors Study", ERIM Report 617507-1-C, 1993.
4. Sampson, R.E., and Wesolowicz, K.G., "Development of an Intelligent Machine for Automating the Forging Process", Proceedings, Sensors Expo 1986, Vol.1, pp.613-629.
5. Wesolowicz, K.G., and Sampson, R.E., "Laser Radar Imaging Sensor for Commercial Applications", Proceedings, SPIE Laser Radar II Conference, Vol.783, Orlando, FL, 1987, p.152.
6. Harding, K.G., and Tait, R., "Moire Techniques Applied to Automated Inspection of Machined Parts", Proceedings, Society of Manufacturing Engineers Vision '86 Conference, Detroit.
7. Air Gage Co., Livonia, MI, 1992 Marketing Information.
8. Besl, P.J., "Active, Optical Range Imaging Sensors", Machine Vision and Applications, Vol.1, No.2, 1988, pp.127-152.
9. Anonymous, Sensor Technology, Vol.8, No.10, October 1992.
10. Olympus America, Inc., Marketing Information, 1993.
11. Marron, J.C.and Schroeder, K.S., "Holographic Laser Radar", Optics Letters, Vol.18, No.5, March 1993, pp.385-387.
12. Walker, J.W., IEEE Trans. Aerosp Electron. Syst. AES-16, 23, 1980.
13. Ejiri, M., "Machine Vision in the 1990s", Machine Vision and Applications, Vol.4, No.2, Spring 1991.

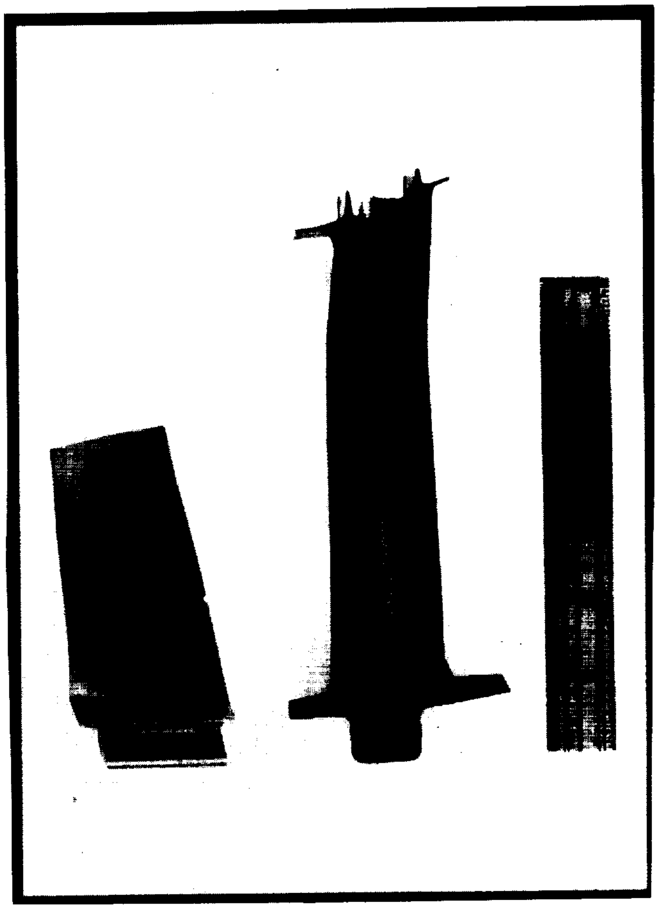


Figure 1. Forged Titanium Compressor Blade (left) and Investment Cast Inconel Turbine Blade (right).

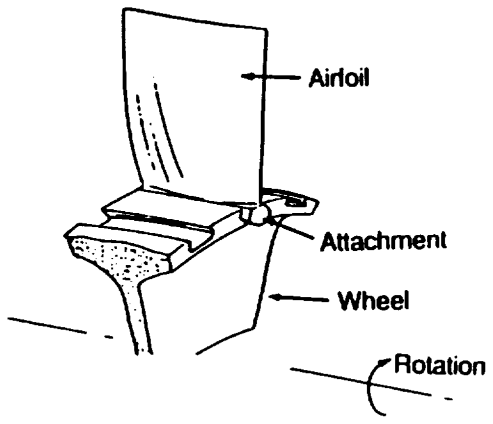


Figure 2. Compressor Blade Features and Assembly.

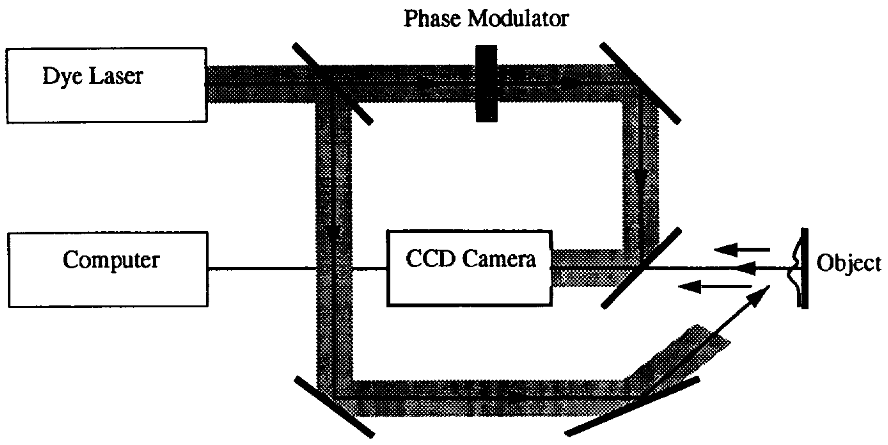


Figure 3. HLR Experimental Setup.

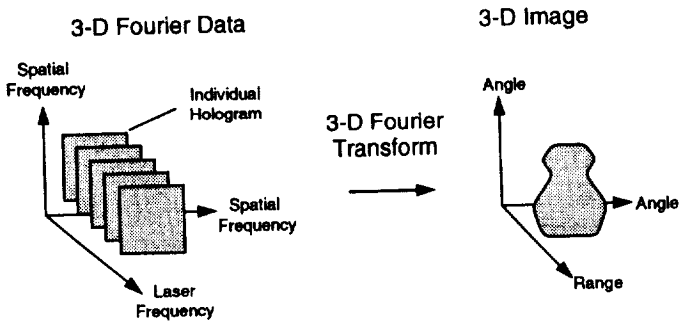


Figure 4. Relationship Between Collected and Recovered Images in 3-D HLR.

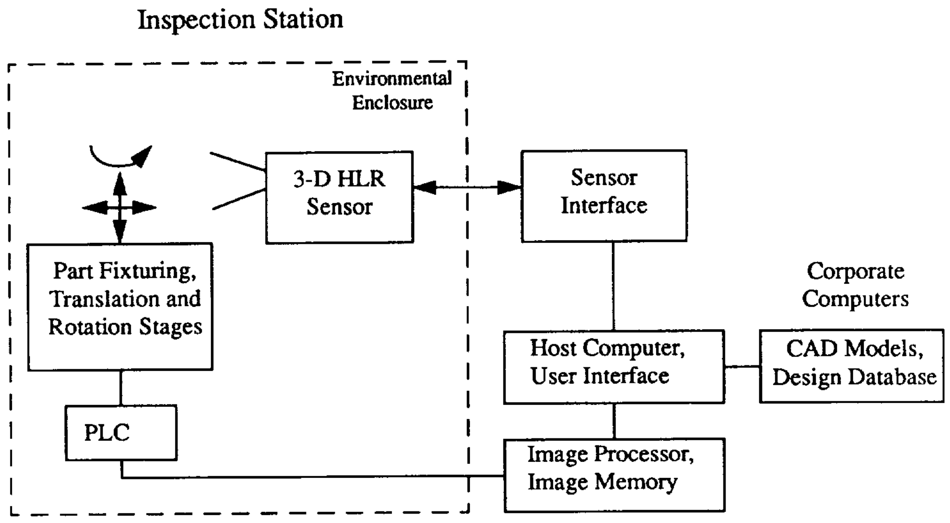

Inspection Station
Environmental Enclosure
3-D HLR Sensor
Sensor Interface
Part Fixturing, Translation and Rotation Stages
Corporate Computers
Host Computer, User Interface
CAD Models, Design Database
PLC
Image Processor, Image Memory

Figure 5. Non-Contact Automated Blade Inspection Workstation Setup.

6071

AIAA-94-1227-CP

TRC RESEARCH PRODUCTS: COMPONENTS FOR SERVICE ROBOTS

N94- 30576

W. Stuart Lob
Senior Engineer
Transitions Research Corporation
Danbury, Connecticut

p. 6

Abstract

Transitions Research Corporation has developed a variety of technologies to accomplish its central mission: the creation of commercially viable robots for the service industry. Collectively, these technologies comprise the TRC "robot tool kit."

The company started by developing a robot base that serves as a foundation for mobile robot research and development, both within TRC and at customer sites around the world. A diverse collection of sensing techniques evolved more recently, many of which have been made available to the international mobile robot research community as commercial products. These "tool-kit" research products are described in this paper.

The largest component of TRC's commercial operation is a product called HelpMate for materiel transport and delivery in health care institutions.

Operation in a Service Environment

Manufacturing operations require precision that is not necessary for mobile robot navigation in most service applications. Service robots generally face situations with less structure and intensity than their counterparts on the assembly line. In a service application, performance is not measured in numbers of rejected parts but in the accomplishment of completed tasks.

In an industrial environment, the robot usually becomes a tightly integrated piece of equipment in the overall manufacturing process. Service tasks such as fetch-and-carry distract and diminish the effectiveness of highly trained personnel. In the service environment, most of the integration is with humans rather than other automated machines. Behavior and interface are important factors in the success or failure of machines working in the service arena.

TRC Technology for Service Applications

LabMate® Mobile Robot Base

The LabMate mobile robot test bed, developed with DARPA support, is now in service at over 100 sites around the world. LabMate is a low cost, mobile robot base designed for use as a component in the development of transport systems and to support research in artificial intelligence, computer science, and robot engineering.

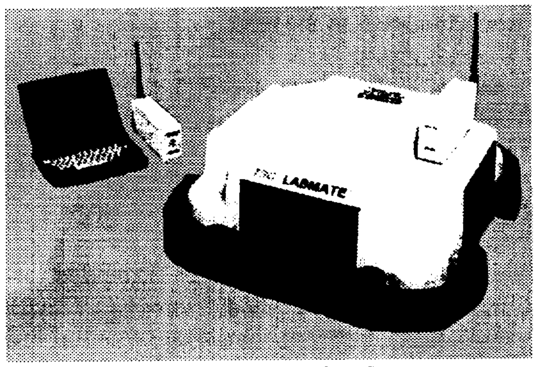


Figure 1 LabMate Mobile Robot Base

The vehicle has a square footprint designed to fit through a standard door opening. Six wheels support LabMate: one passive caster at each corner, and two driven wheels centered longitudinally along either side. Each drive wheel is under individual servo control. The only moving parts on the vehicle are the fixed drive wheels. Differential variation of wheel velocities steers the vehicle. When each wheel is driven at identical velocities in opposite directions, the LabMate spins in place. This capability to change orientation without translating position is important for sensor systems with a limited field of view. The differential steering architecture eliminates the need for an additional rotating turret.

The drive motors are servo controlled by a microcomputer controller based on the Motorola 68HC11 microprocessor. The computer monitors and controls wheel position and converts the rotational displacements of the wheels to a position and angle in a two-dimensional Cartesian space, i.e. X, Y, and Θ expressed in millimeters and degrees/100.

Through the RS-232 interface, the host application can query the LabMate at any time to determine position and orientation and issue motion commands.

Payloads as large as 200 pounds can be mounted on board. Payloads typically include computers, manipulator arms, communications gear, cameras, and sensors.

The low profile of the LabMate base provides an important advantage. Active, awkward payloads with high moments of inertia affect vehicle stability. To minimize these effects, the LabMate drive hardware and batteries are located within a few centimeters of the floor.

LabMate serves as the foundation for the TRC HelpMate® autonomous service robot and several similar systems developed by TRC customers.

LightRanger™

The TRC LightRanger Light Direction and Ranging (LIDAR) system delivers fast low noise range information from an actively scanning eye-safe infrared beam. LightRanger locates oblique surfaces missed by acoustic techniques; specular reflectivity is not required.

The LightRanger projects the beam from an infrared LED and continuously sweeps the beam 360° through a volume of rotation 45° across the cross section. A large area lens gathers reflected light from objects, and on-board circuitry then compares the phase of the modulation of the returned signal with that of the transmitted light. A built-in 68HC11 microprocessor converts this information into true range units and transmits it to the host computer via an RS232 serial link or Ethernet.
The sweeping and nodding mechanism is driven at up to 600 rpm by one servo motor. This translates into a 10 Hz refresh rate for the entire circumference of the observed volume. In operation, LightRanger can locate white objects out to 10 m and darker objects (such as blue denim) to 7 m.

The light-based direction and ranging system differs significantly from traditional vision systems. A vision system acquires an entire frame of data represented by the image sensor plane. The image must then be analyzed to extract feature information and indirectly compute the distance to obstacles within the image field. The LightRanger generates its three-dimensional map with a scalar measurement under active, mechanical servo control. The scalar reading from each optical sample is combined with the instantaneous heading and elevation of the scanning mechanism to yield a position in three-dimensional space.

The vertical nodding action of the scanner is deliberately set to a fraction of the rate of the horizontal scan. The sensing beam traces a series of flat spirals that approximate horizontal planes. The effect is to produce two dimensional maps of the environment at several elevations. This provides fine resolution data on the planar location of obstacles and targets, and course resolution data on elevation, which is optimal for a mobile robot navigating in a two-dimensional horizontal plane.

LightRanger Beacon Navigation System

The Beacon Navigation System (BNS) automatically senses and reports $[X,Y,\Theta]$ position and heading at ranges up to 25 meters within a quadrilateral area defined by four retroreflective beacons. BNS, which is based on the TRC LightRanger, acquires and locks on to the four beacons during a stationary initialization sequence lasting a few seconds. From that moment on, BNS sends a continuously updated stream of $[X,Y,\Theta]$ information that a host computer or track following mechanism can use to control the vehicle trajectory.

BNS will continue to operate even if two of the beacons are completely obscured. If a beacon is moved after initialization, BNS software will automatically adjust the bounding quadrilateral to compensate.

The sweeping motion is driven at up to 60 rpm by one servo motor. This translates into a 1 Hz refresh rate for the entire circumference of the observed plane.

BNS was developed for commercial floor sweeping operations in large, open areas. The BNS light direction and ranging unit operates at a lower wavelength than the stand-alone LightRanger. This increases the range but reduces the precision of the data. This is an appropriate trade-off: the LightRanger is designed for navigation in comparatively crowded corridors where clearances for passage are measured in centimeters, while the BNS is designed to control cleaning machines wide paths in open spaces up to 30 meters per side.

SonaRanger™

The SonaRanger senses its environment by bouncing ultrasonic pulses off objects and timing the delay before the reflection is detected. The system consists of a small microprocessor that controls sonic actuation and detection of up to 24 ultrasonic transducers.

The ultrasonic transducers transmit signals covering a 15° cone out as far as 10 meters. On a mobile robot vehicle, the sensors are aimed forward for obstacle detection and landmark identification, to the sides for detecting wall surfaces, and along vertical axes for detection of obstacles with suspended horizontal surfaces such as tabletops and desks with overhanging ledges. The sensor data are continually monitored and verified through a number of filtering techniques, such as comparing successive readings from the same sensor and summation of readings.

Logarithmic-Polar Vision

Image analysis and compression techniques based on logarithmic-polar mapping were introduced in the 1970's. In the past five years, functioning prototype logarithmic-polar vision guidance and control systems have been demonstrated by researchers at TRC and a number of universities around the world.

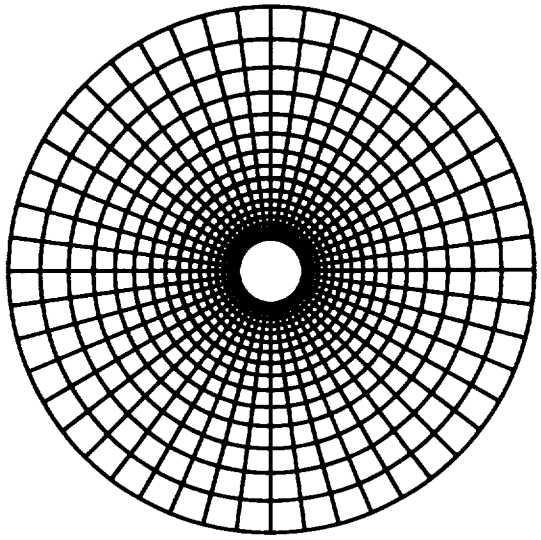


Figure 2 The Logarithmic Polar Coordinate Space

In logarithmic-polar representation, image plane pixels are arranged in a polar coordinate system where the distance between the concentric circular pixel boundaries grows exponentially.

In the logarithmic-polar representation, pixel count drops by nearly two orders of magnitude. The image has high resolution in the center and low resolution at the periphery. Images in this coordinate system are invariant in rotation and zoom. In the Cartesian space, a matrix transformation on each of the thousands of individual pixels is required for rotation and zoom. These operations in the logarithmic-polar space are accomplished by merely shifting the image data. This dramatically reduces the computational requirements for image transformation. In logarithmic-polar representation, the computer processor accomplishes these transformations quickly by shifting a single block of memory across the address space. In a frame buffer, this can be achieved by indexing all addresses by the shift vector.

The application of a Hough transform to the logarithmic-polar image yields edge and line information that can be used for identification of objects in the environment.

There is a growing body of evidence that human and higher mammalian vision processes images in the logarithmic-polar space. Indeed, the arrangement of cones on the retina of human eyes provided the original inspiration for using the logarithmic-polar space.

The highest resolution and detail in a logarithmic-polar image are always at the center of the image. A shift of attention or displacement of the imaging device requires rapid and accurate repositioning of the direction of gaze. This is a highly refined capability in even the most primitive animals.

The bulk of research in vision to date has concentrated on image analysis techniques where the camera position is fixed or rigidly attached to a moving vehicle. More recently, vision systems where the camera is actively controlled as part of the imaging process have appeared in research laboratories. Instead of intensively processing the entire image delivered by a camera at an arbitrary position, these new active vision techniques use the direction of gaze as an integral part of the algorithm.

A vision system based on the principles of logarithmic-polar space and a single camera requires active camera servo control to keep the image centered on the object of interest. With two cameras, the differences between each image can be used to extract depth information. Binocular active vision is rapidly emerging as a premier sensory modality for robot navigation and obstacle avoidance. Pioneering work by researchers at NIST, University of Genoa, Florida Atlantic University and others have shown optic flow and binocular vision can provide rich, 3-D perception at high speeds. For robot vehicles, the vision sensor mount must be steerable to compensate for vehicle motion, lock on to targets of interest, and converge for binocular stereo.

TRC has developed a high performance light-weight binocular vision head for robot vision. Both speed and precision are important. Trademarked the "BiSight," this system mimics the articulation, speed, and precision of human vision. Two CCD cameras are mounted on direct-drive brushless DC motors to provide saccadic (fast motion to new point of interest) motions at speeds of up to 1,000° per second, and binocular vergence motions at precisions of a fraction of a degree. Vergence and tilt axes pass through camera nodal points, assuring a constant binocular baseline throughout the range of motion, and complete separation of camera rotation and translation.

TRC is developing advanced image processing algorithms for NASA based on Gabor functions, to give mobile robots high precision 3-D perception of moving environments.

The Opportunities in Health Care

Several years ago, TRC identified a need in health care institutions for improved transport systems. As hospitals, clinics, and other health care institutions have grown, single buildings have expanded into sprawling campuses. In a typical expansion project, many departments are relocated to increasingly remote locations. Trips to a department formerly a couple of doors down the corridor become half-hour excursions.

Trained specialists such as pharmacists and nurses achieve zero productivity in their respective fields when they are walking across the campus to retrieve a sample or deliver paperwork.

Fixed, "hard-wired" transport systems such as pneumatic tubes were developed as an early solution. These systems were expensive to install, difficult to maintain, and costly to modify as the institution grew. Reprogramming meant ripping through walls, tearing up floors, and rewiring switch panels. TRC has seized the opportunity to develop a low cost robot that requires minimal facility modification, is easily maintained or replaced, and can be reprogrammed with a simple CAD drawing.

HelpMate ®

The HelpMate trackless robotic courier is TRC's principal product. HelpMate transports supplies between remote locations in office and institutional environments without a dedicated guidance system. It reduces or eliminates courier trips by skilled hospital staff.

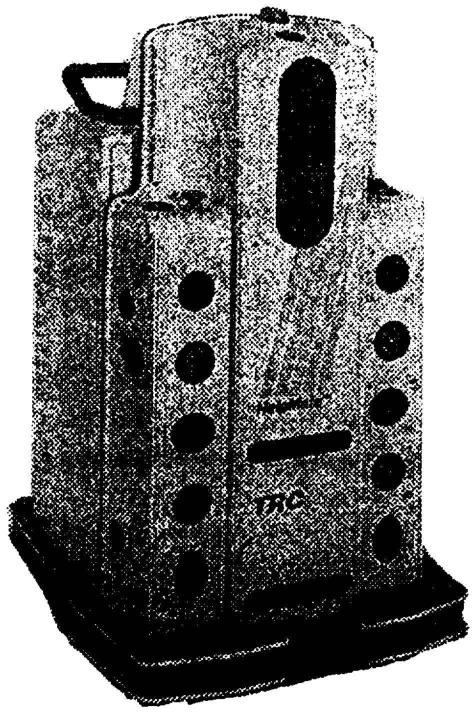


Figure 3 HelpMate Autonomous Robot

HelpMate navigates fully autonomously using passive data from its environment. This eliminates expensive facility modifications to install guidance aids, such as a network of beacons or embedded wiring in the floor or ceiling.

The HelpMate is the ultimate application for most of the technology developed at TRC. HelpMate is controlled by a main on-board processor with smaller peripheral processors for each sensor system and the drive carriage. As it moves down a corridor, HelpMate uses an ultrasonic sonar ranging system to measure distances to walls and potential obstacles. A vision system that uses twin, parallel planes of infrared light locates obstacles to the front of the vehicle. The data from all the sensors is collected by the central processor and placed into a stored map of the local environment. The central processor analyzes the map and calculates the path, avoiding obstacles as needed. Other processors handle the user control interface, indicator lights, compartment latches, and communications with the central fleet manager computer, elevator controls, and delivery enunciators.

An advanced prototype of the HelpMate equipped with a LightRanger is now being used to develop improved obstacle avoidance and navigation algorithms. Walls, stationary objects, and moving obstacles can be detected and tracked with greater resolution.

TRC is working with NASA to produce a demonstration vehicle consisting of a LabMate mobile robot base integrated with a binocular logarithmic polar vision system. This vehicle will be used to develop new techniques for guidance, obstacle avoidance, and object detection and recognition.

Bibliography

1. Engelberger, J. F. Robotics in Service, MIT Press, Cambridge, 1989.

2. Evans, J. M., Krishnamurthy, B., Pong, W., Skewis, T., Barrows, B., King, S., Weiman, C. and Engelberger, G., "Creating Smart Robots for Hospitals", *Proc Robots 13 Conf*, Wash. D.C., May 1989.

3. Krishnamurthy, B. et. al., "HelpMate: A Mobile Robot for Transport Applications", *SPIE Vol. 1007 - Mobile Robots III*, 1988.

4. Miller, G. L. and Wagner, E. R., "An Optical Rangefinder for Autonomous Robot Cart Navigation", *SPIE Vol. 852 - Mobile Robots II*, pp. 132-144, 1987.

5. Skewis, T., Evans, J., Lumelsky, V., et. al., "Motion Planning for a Hospital Transport Robot", *IEEE International Conference on Robotics and Automation*, April 1991.

6. King, S. J. and Weiman, C.F.R., "HelpMate Autonomous Mobile Robot Navigation System", *SPIE Vol. 1388 - Mobile Robots V*, 1990a.

7. Sandini, G., and P. Dario, "Retina-Like CCD Sensor for Active Vision", *NATO Advanced Research Workshop on Robots and Biological Systems*, Il Ciocco, Tuscany, Italy, June 1989.

8. Schwartz, E. L., "On the Mathematical Structure of the Retinotopic Mapping of

Primate Striate Cortex", *Science*, Vol 277, page 1066, 1985.

9. Weiman, C. F. R. and G. Chaikin, "Logarithmic Spiral Grids for Image Processing and Display", *Computer Graphics and Image Processing*, (11), pp. 197-226, 1979.

[illegible]

AIAA-94-1228-CP

N94-30577

An update on "Lab Rover": A Hospital Material Transporter

[illegible]

P. 2

PAUL MATTABONI
President
Cyberotics, Inc.

ABSTRACT: An update on "Lab Rover", a hospital material transporter current health care costs in the USA, are 1% of the G.N.P. This translates to 750 billion dollars/year. By the year 2000, health care costs are projected to reach one trillion dollars/year or 20% of the G.N.P. Health care costs are skyrocketing and the Government has made cost containment its number one priority for health care. Cyberotics' approach to cost containment has been to automate material transport within medical institutions. Conventional material transport now utilizes people power, push carts, pneumatic tubes and tracked vehicles. Hospitals are faced with enormous pressure to reduce operating costs. Cyberotics, Inc. developed an Autonomous Intelligent Vehicle (AIV). This battery operated service robot was designed specifically for health care institutions. Applications for the AIV include distribution of clinical lab samples, pharmacy drugs, administrative records, x-ray distribution, meal tray delivery, and certain emergency room applications. The first AIV was installed at Lahey Clinic in Burlington, Mass. Lab Rover was beta tested for one year and has been "on line" for an additional 2 years.

INTRODUCTION:

During the past 18 months, Cyberotics embarked upon a program that allows for manufacturing cost reduction, expanded intelligence, navigation enhancement, and improved appearance. This resulted in Cyber V, the latest achievement in a technology which represents over 10 years of research and development.

OPERATION:

The vehicle's motion and steering is provided by a velocity controlled, differential wheel drive. Main power for all systems is supplied by a pair of high capacity batteries. The vehicle operates for a minimum of an eight (8) hour period. The power package is designed for easy replacement .

The "on board" computer works in conjunction with ultra-sonic and infrared sub-system, to navigate freely throughout the work environment. The AIV does this without the use of wires, or floor tapes. It is completely flexible and can be programmed to navigate a defined delivery route. It is capable of accepting instructions, manually through the control panel keyboard, or through a digital radio communications link. Additionally, a complete operational status can be acquired through the RF link.

Infrared beacons at key locations communicate to the vehicle, instructing it to stop, turn, or sound an audible arrival signal to area personnel. If there is no one to unload the vehicle, it "times out" and, again continues with its assigned tasks.

MATERIAL HANDLING:

The work surface of the vehicle may be optionally divided into spaces for trays, racks, or boxes. With the addition of the radio frequency (RF) communications option, the vehicle may be used in a "dispatch mode". This allows for continuous scheduling from a central base station, which may also be connected to the hospital's Local Area Network (LAN). This option also allows for instant vehicle location information, at any time, from any work station. This dispatch
system handles multiple vehicles.

SALIENT FEATURES:

The AIV's intelligent software and its ultra-sound navigational system combine to allow safe mobility through corridors, avoiding people and other unforeseen objects in its path. Top speed is approximately 2.5 mph, with automatic slow-down while avoiding obstacles. A tactile bumper is an added safety feature, which allows for an instant stop and automatic assessment of unusual circumstances.

A brief review of Cyberotics' sensor and navigation development.

Cyber I Feasibility Model

The original design consisted of a circular array of ranging transducers comprised of two inch speakers, a separate transmitter and receiver, having four speakers each. The Polaroid transducer was not available yet.

The speakers have a natural resonance of 5 KHZ and when set in an array of four speakers with a backing plate, it provided a beam width of 36 degrees. An eight bit 16K OSI computer provides all the computing power. While crude by today's standard, it indicated that the approach had promise.

One particular problem to overcome is in the acoustic vision. Elimination of false echoes, mirror effects and cross talk were of particular concern. An open loop wheel drive, a natural language processor, and a video screen was used. The control software was based on behavior reaction. It was functional, but hard to add new behaviors, because the control parameter was embedded in the main body of the software. A combination of Assembly Code and Basic was used.

Cyber II

This design was to be the answer to the personal robot craze of the 1980s, but was too late and too expensive for the hobbyist and experimenter. The circular transducer array proved to be a viable concept.[1] It provided a geometry that all but eliminates the specular effects, by insuring that a transducer was always perpendicular to a vertical surface.

The Polaroid transducers were now available. This provided a more efficient way of collecting range information. Running at 50 KHZ with a 22 degree beam width, it allowed for a higher resolution sensing of the environment. A pulsing pattern was devised to help eliminate false echoes. The transducer density, however, was not sufficient enough to eliminate blind spots in its vision. It operated quite well for the experimental use, but was not reliable for practical applications. A new programming concept was developed - behavioral induction.

In behavioral induction, a model of behavior is provided. The model consists of a set of co-efficient in a data base, that are selected to set the parameters of a fixed function. The model is compared with real world data. A difference or error representation is derived. A set of heuristic rules evaluates the error and specifies appropriate wheel responses. Different functions, coefficients, rules, and responses are capable of producing a wide range of behavior types. Automatic and fixed selection of behavior types, allow the vehicle to move freely through the environment.

The ability to behave and to react to the environment, does not necessarily give the vehicle a useful purpose. The ability to go from point A to point B, requires navigation and command input to give it direction.

A reference position system was added that provided information to the vehicle when it reached key locations. The system consisted of an ultra sonic ranging transducer which measured the height of an object overhead. Stations were built with dimensions of specific heights. When height was detected, that location was fixed, e.g.: ceiling heights, doorways, stations. A command at the keyboard would be mapped to a location having a specific height, thereby, fixing the robot at that location. This system was convenient to implement, but not meant to be practical. It's use was to assist in developing the navigational software.

Cyber III

A closed loop velocity controlled servo drive was added. This provided a selection of 255 speeds. The motion of the vehicle could now be contoured for a smooth response.

The acoustic overhead detector was replaced by an infrared beacon system. The basic navigation concept remains the same. The infrared beacons provide a positive method of location without ambiguity. It also provides directional information.

The navigation algorithm is goal seeking. At each beacon, the vehicle path is compared with its intended destination. If a correction in path is required, the navigation algorithm reevaluates and chooses a new path. Cyber III contained a total of 60 Polaroid transducers. This provided for higher resolution for the vision system, but introduces acoustic problems. Transducers are now closer together and isolation becomes a problem. A baffle was added to serve several purposes. It isolates the output of one transducer from the input of another. It reduces the problem of false echo arrivals by limiting angular reception. It also eliminates side lobes produced by the transducers.[2]

Cyber IV

Product Enhancement:

This model was intended to be product ready, requiring all of the mundane, but necessary, features to make it practical. As an addition to the goal oriented navigation, an error correction system was found to be necessary. Occasionally, the vehicle would be diverted from its defined operating area and wander down the hallway where no navigation beacon exists. To alleviate this problem, an error correction system was devised by utilizing peripheral beacons specifically for recovery purposes. These beacons are placed at edges of the defined work environment, in areas where the vehicle is not allowed to go. They provide explicit instructions on how to get back to the defined area.

Additional features that were added are:

1) Smart bumpers
2) Emergency shut down switch
3) Go - No go button
4) Servo disable switch
5) Battery gage
6) Key lock on - off switch
7) Battery removal cart

Cyber IV was developed as an Autonomous Intelligent Vehicle (AIV), for the hospital market; and beta tested (1991) at the Lahey Clinic of Burlington, Mass., where it was promptly and affectionately named "Lab Rover." After beta testing was completed, Lab Rover has continued to remain "in service" for over two years, delivering biological samples to various labs, proving that a system based on this technology can be reliable and cost effective.

Cyber V

While Cyber V required no new major concepts, a range of enhancements, however, have been incorporated making the difference in terms of becoming product ready.

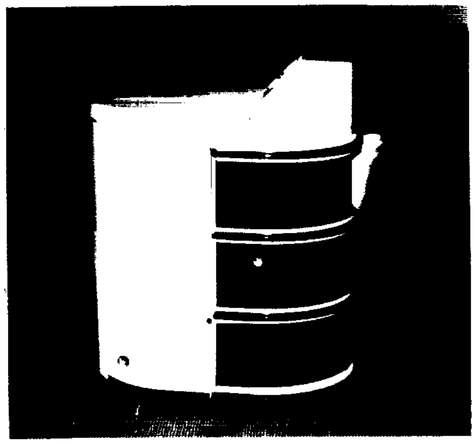
A major engineering effort was undertaken to reduce manufacturing and servicing costs, replacing expensive machined parts with one piece formed metal and molded plastic parts. An outer fiber glass shell allow for improved cosmetics and easy access to internal circuits. Pneumatic tires were replaced with solid tires, eliminating a nagging flat tire problem. Gel cells replaced liquid lead acid batteries. Gel cells, while more expensive initially, reduced battery service calls to zero. This eliminated the need to add water to the batteries on a monthly basis resulting in a net savings in service costs.

Particular attention has been placed on the human robot social interaction. The vehicle operates in the same space that people do. While people have developed a protocol for working in small spaces, they demand equal access and respect. Robot vehicles must do the same. They must be perceived as important contributors in the work environment. A timid robot vehicle will not gain respect. e.g.: When elevators were first introduced, people would hold the doors open, denying use to users on other floors. This presented an operational problem to elevator manufacturers.

A solution to the problem was to modify the elevator door controls so that they would automatically close after a prescribed amount of time. More aggressive doors would actually push people out of the way. The publics initial reaction to this modification was annoyance, but soon gained acceptance as a necessary behavior modification. We have found that adjusting the behavior of the robot vehicle to politely approach, but come very close to people blocking hallways, soon gained respect for the robot vehicle and people now, automatically, move out of its way.

Conclusion:

We believe that Cyber V is now ready for the market place and are now installing Cyber V systems in various applications. We expect that new problems may arise and solutions must be found. This technology is now ready for the commercial market place and are now shipping systems to customers and licensed OEM dealers.



LAB ROVER - CYBER V

References:

(1) P. J. Mattaboni - Patent No. 4638445 January 20, 1987
(2) P. J. Mattaboni - Patent No. 5165064 November 17, 1992
(3) J. Haugeland - "Mind Design" The MIT Press, 1981
(4) A. Barr & E. Feigenbaum - "The Handbook of Artificial Intelligence" 1981, Vol. 1
(5) P. H. Winston - "Artificial Intelligence" - 1977
(6) T. Marrs - "An Introduction to Cyber I" The Personal Robot Book - 1985
(7) P. J. Mattaboni - "Lab Rover - Hospital Material Transport Robot" The proceedings of The International Robots & Vision Conference April 1993

L5-13

AIAA-94-1229-CP

N94-30578

A Robotic Wheelchair

David P. Miller
The KISS Institute
10719 Midsummer Drive
Reston, VA 22091

Edward Grant
Power Concepts Incorporated
5030 E. Jensen
Fresno, CA 93725

-52-63
21675
p. 5

Abstract

Many people who are mobility impaired are incapable, for a variety of reasons, of using an ordinary wheelchair. These people must rely on either a power wheelchair, which they control, or another person to push and guide them while they are in an ordinary or power wheelchair. Power wheelchairs can be difficult to operate. If a person has additional disabilities, either in perception or fine motor control of their hands, a power chair can be difficult or impossible for them to use safely. Having one person push and guide a person who is mobility impaired is very expensive, and if the disabled person is otherwise independent, very inefficient and frustrating. This paper describes a low-cost robotic addition to a power wheelchair that assists the rider of the chair in avoiding obstacles, going to pre-designated places, and maneuvering through doorways and other narrow or crowded areas. This system can be interfaced to a variety of input devices, and can give the operator as much or as little moment by moment control of the chair as they wish.

1 Introduction

The powered wheelchair as an assistive device for the mobility impaired is a direct outgrowth of the basic metal tube parallel frame design philosophy that originated just before W.W.II. It was developed by adding DC drive motors to the manual design and an analog differential joy stick for direction control. In many cases, speed control as an on-off-coast function with little or no progressivity. Late in the 1970's, the advent of computer miniaturization led several designers to investigate the potential applications of digital control as means of expanding range of capability, user features and environmental compatibility. While the fruits of these previous efforts are just now beginning to enter the marketplace, all are flawed in that they lack the sort of "intuitive" directional capability commonly exercised by the able bodied when proceeding from point A to B. Although this is not necessarily a major problem for the mobility impaired individual who retains adequate upper body and extremity motor control, for those with more profound loss and/or multiple disabilities, it can result in near or total removal of personal options for independence.

1.1 Current State of the Art:

Microprocessor-controllers are now available with varying degrees of capability and programmability. Much of the effort, to date, has focused on providing clinicians the ability to "program" performance parameters, improve the linearity of control/speed response and develop chair to "external" environmental interfaces. The rate of acceleration and turning are tuned to a particular user's capabilities and environment.

Quest technologies provided a degree of automation for its *access* chair that related to edge and drop-off recognition, which is probably the only FDA "approved" use of automation in wheelchair applications. Those that would benefit from the application of more automation in chair control include:

- Upper level spinal cord injured incapable of operating joy stick controllers. Such individuals currently use either a chin adapted joy stick, head controller, or a "sip" and "puff" actuator.
- Neurologically impaired (stroke, cerebral palsy, ALS, MD, MS, etc.,).
- People with low and eccentric vision.
- Individuals with multiple handicaps.
- Geriatric populations with declining physical abilities.

Despite the advances in robotics and AI research in other fields, little practical work has been done in adapting power wheelchair control to be more usable by the class of potential users outlined above. What work has been done (e.g., [3]) uses customized platforms and electronics and is prohibitively expensive.

1.2 An "Autonomous/Intuitive" Controller:

An autonomous controller should embody the capabilities necessary to safely and efficiently operate a powered wheelchair for a wide variety of individuals with profound motor and neurological control functions. It should be able to track a given course from A to B while avoiding intervening obstacles as part of its decision making process, rather than that of the operator. The ability to perceive unsafe environments should be incorporated as some of the target user population is so positioned or otherwise limited that their range or degree of effective vision is severely circumscribed. Essentially, it should be possible for its user to operate the system using various control interfaces that range from a joystick through chin and "sip" and "puff" to voice and eyegaze. All operating parameters (speed, turn rate, access to options, etc.,) must be readily prescribable and programmable by the clinical "intervention" team of doctors and therapists to assure professional acceptability.

The remainder of this paper describes Tin Man, a *Vector* brand power wheelchair which has an enhanced controller and sensor array. Tin Man allows the user to operate the chair in a variety of modes ranging from normal power chair operation through simply designating a heading which the chair will follow while automatically skirting obstacles. But perhaps the most significant accomplishment of Tin Man is that it involves virtually no custom electronics or mechanics. All components are consumer off the shelf, and the component cost of the modifications to the standard power chair are less than $500, and take less than a day to put together and install on the chair. The initial design of the controller and construction of the software took appreciably longer.

2 System Design

This section describes the hardware and software of *Tin Man* the robotic wheelchair.

2.1 Hardware Configuration

Tin Man is built on top of a commercial pediatric wheelchair from Vector Wheelchair Corporation. In its current instantiation, Tin Man has no electrical interface between the chair's controls and the robot's computer. Instead, there is a mechanical interface. The control computer controls two servomotors which are mechanically linked to the standard joystick that comes with the chair. The user enters their commands through an input device (usually another joystick). The commands and sensory data are processed by a commercial micro-controller based around the Motorola 68HC11 processor. The micro-controller then commands the servo motors which move the main joystick on the chair. The joystick position is read by a standard wheelchair analog controller which generates PWM signals to the two drive motors.

Tin Man has five types of sensors:

- Drive motor encoders;
- Contact sensors;
- IR proximity sensors;
- Sonar rangefinders;
- Fluxgate compass;

Tin Man is equipped with encoders on each of its drive motors. The drive motor encoders, after gearing, deliver a resolution of 6.725 tics per inch. With the encoder resolution and the robot's wheel separation, theoretically the robot's orientation can be known to a resolution better than 0.01 radians. Unfortunately, because of the width of the drive wheels, slippage, wheel distortion, etc., it appears that the robot is only able to turn within $\pm 10\%$ of the commanded amount. As a result, dead reckoning errors can grow quickly.

There are eight contact sensors on the robot. Each sensor is made from a resistive strip approximately ten centimeters in length. As the strip is bent, its resistance changes, and the degree of the bend can be calculated from the current flow through the strip. Two of the strips are mounted on each side of the robot, one in front of the wheel and the other in front of the armrest. The remaining four sensors are mounted on the front. These sensors are enclosed by a sheet of foam rubber. The foam fills the gaps between the sensors. If the foam contacts an obstacle, its shape is distorted causing the sensing strips to bend.

There are four IR proximity sensors distributed evenly along the front and sides of the robot. These sensors emit a coded beam of infrared light. If an object is nearby, the light is reflected back to the sensor. When a reflection is detected the sensor goes high. These sensors are very albedo sensitive.

There are six sonar rangfinders on Tin Man. Each sonar has a resolution of one centimeter, a minimum range of thirty-five centimeters and a maximum range of five meters. It takes each sensor approximately two-hundred milliseconds, from the time it is activated until it settles on a reading. Due to port limitations, all of the sonars are ported into the same timing port. They are sampled round robin. Each sonar can be activated or deactivated in software, and only the active sonars are polled. If all the sonars are active, it can take over one second between readings from a specific sonar.

The fluxgate compass is a standard compass meant to be used in an automobile. The coils that control the display are directly wired to two of the analog to digital ports on the micro-controller. The computer can distinguish changes in heading of approximately ten degrees. While not adequate for accurately traversing long, open distances, this is sufficient resolution for navigating along streets and in building corridors where the environment can help you keep on course.

2.2 Software Design

The software for Tin Man is written in IC, an interactive, multi-tasking dialect of the C language. Each sensor type has its own asynchronous process which monitors those sensors. With the exception of the sonars, every sensor is polled at at least 5Hz. The maximum safe speed of the chair is governed by this sensor refresh rate combined with the deceleration rate of the chair.

All the sonars are multiplexed through a single port and into a single timing register. It takes several ultrasonic pulses to ensure a reliable distance reading from the sonar, and from the time the first pulse starts, till the last echo returns, a single sonar owns the timing register. A single sonar can be read at 3-5Hz. Most modes of the robot use at least three active sonars leading to an update rate of approximately 1Hz.

In the manual operation mode, the operator gives their input through a joystick. The micro-controller reads the joystick and issues servo-motor commands to cause the chair's joystick to copy the movements of the operators joystick. There are three semi-automatic modes that Tin Man can run. They are:

- Human guided with obstacle override;
- Move forward along a heading;
- Move to X,Y.

In all three modes, the same priority scheme holds true:

1. If a contact sensor reads true, the chair moves away from the point of contact;
2. If a proximity sensor reads true (and contact sensors do not) then the chair turns away from the direction of the sensor reading true (if both front sensors read true then the chair will back up, if both side sensors read true then the chair will go straight, slowly);
3. If a sonar senses an obstacle less than 60cm away in front or behind then the chair will not move forward or backward. If a sonar senses an obstacle less than 1m away, then the chair will turn away from the direction of the obstacle;
4. The robot follows the designated heading or towards the designated waypoint, unless this conflicts with one of the sensor rules listed above;
5. The chair follows thè commands from the user input device. unless the commands conflict with one of the rules above.

When operating in the obstacle override mode, the chair follows the user's instructions except when a nearby obstacle is detected. When an obstacle is detected, the chair will modify its heading, following the a safe heading that is as close as possible to the heading being input by the user. If the user puts in a stop, the chair will stop. This is probably the most common mode to run the chair. It is especially useful when training someone to use a power chair. It is also helpful when maneuvering in tight spaces or through narrow doorways. For an operator with slow reflexes or limited perception, this mode allows the chair to be operated at a speed much faster than would otherwise be safe. In all cases, it greatly reduces the risk of impact with an obstacle, and the severity of an impact should one occur.

The move forward along a heading is the mode that is most useful for someone who has a very limited amount of bandwidth for input to the chair. The chair can be spun until the desired heading is reached. When at the desired heading, the chair moves forward, avoiding or maneuvering about obstacles as needed. If the chair is pointed in the general direction of a doorway, it will autonomously maneuver through the doorway. If pointed down a hallway, the chair will continue down the hallway until blocked. The only control needed by the user is to: put the chair into this mode; designate the proper heading; tell the chair when to stop. Currently all three commands are executed by pressing a button at the desired time, but they could as easily be commanded by monitoring eye blinks or a sip/puff controller.

The move to X,Y position mode allows the user to specify a specific position in absolute coordinates for the chair to go to. A heading to the desired point is calculated, the chair turns to that heading and then moves forward much as in the previous mode. Obstacles are avoided, and after each deviation, the chair heads straight for the goal location. This mode is meant to be used only in situations where there is a mostly clear path towards the goal location. To go to locations that involve going around corners, down corridors, etc., it is best to input a series of locations representing waypoints for the robot to follow.

3 Future Work

Tin Man has two major shortcomings that prevent it from being a useful device for the mobility impaired: the current user interface and the current handling of raised obstacles such as tables and desks.

The current user interface is all run through a joystick and menu with two selector buttons on the micro-controller board. In order to switch between modes, or set specific X,Y positions, a level of dexterity, visual acuity, and flexibility is required that is inconsistent with the targeted user group. These problems can be easily overcome by repositioning the control panel on the chair's armrest, using larger buttons and a larger, backlit display.

A more serious shortcoming is that the vast majority of obstacle sensors are located near to the ground, where the vast majority of obstacles are to be found. However, common objects such as tables and desks, which may have clearance adequate for the chair, do not have adequate clearance for the user. We believe that an upward looking sonar would be able to detect when the chair is starting to go under an object without adequate clearance for the user. When this condition is detected, appropriate action could then be taken by the micro-controller. Stairwells and other dropoff could in principle be detected similarly by using a downward looking sonar or proximity sensor.

We plan to supplement the chair's current capabilities (obstacle avoidance while following user commands, following a heading, or going to a specific point) with the following:

Backtracking: the chair would retrace its previous movements up to some limit or till stopped by the user. This would allow the user to quickly and easily return to a previous location or room. This would be accomplished by recording waypoints every time the chair changed its heading significantly and then automatically performing a series of X,Y moves to the list of waypoints, in reverse order.

Wall Following: the chair would align itself to the wall (selected by the user) and move along that wall at a constant distance (while avoiding obstacles) until terminated by the user. This would be implemented by servoing (when no other obstacles were closer) to a preset distance on the side sonars.

Docking: the chair would approach an object in front, slow down and stop at first contact. If the object was a table or a desk, the chair would slow and then stop when it was a prespecified distance under the object.

Automated Safing: these functions would prevent the chair from moving too quickly over bumpy surfaces or going over terrain that might cause tipover. Both functions could be implemented using roll and pitch "3-position" sensors.

Path Planning: the Tin Man micro-controller can easily be connected to a general purpose computer for carrying out more complicated tasks. The capabilities currently implemented on the chair can act as the low-level reactive skills for an autonomous agent architecture that has been created [4, 2, 5, 1, 6]. Under this mode, the user would interface through a laptop or similar additional computer installed on the chair, hooked into the chair's micro-controller. The laptop might have a CAD model of the building. The user would specify where the chair currently is, and where the user wants to go. A topologic path planner would use the model of the building to generate waypoints for the controller. It could also monitor some of the sensors to update its position during the traverse (e.g., monitor the side looking sonars so that it would know when it had moved through doors). This way, dead reckoning errors could be kept to a minimum. If the chair should stray too far due to slippage, the user could update their position on the map. The laptop could be used to drive a host of more sophisticated interfaces (than the joystick and buttons) including an eye tracker, a speech interpreter, or a menu driven "sip/puff" controller.

4 Conclusions

We have constructed a robotic wheelchair that is capable of maneuvering through a wide variety of typical environments without collision. The chair takes direction from the human user in a variety of forms ranging from direct control to destination specification. This type of chair should prove useful to persons with mobility impairment and limited visual acuity, spasticity, diminished fine motor control or any condition that makes it difficult for them to independently operate a normal power wheelchair.

The most significant accomplishments of this project are: the equipment and parts are all readily available and off the shelf; the cost for the modifications represent only a 10% increase in cost over a normal power wheelchair. Tin Man is an existence proof that robotic aides for the mobility impaired do not have to be prohibitively expensive.

Acknowledgments

The authors would like to thank Marc Slack, Anne Wright, Mark Westling, and Mike Wessler for their help in developing Tin Man. This work was supported in part by The American Association for Artificial Intelligence, Apple Computer Corporation, The Jet Propulsion Laboratory under a grant from their Director's Discretionary Fund, The KISS Institute, The MIT AI Laboratory, The MITRE Corporation, Power Concepts Incorporated, and Vector Wheelchair Corporation.

References

[1] C. Elsasser and M. G. Slack. Planning with Sequences of Situated Skills. In *Proceedings of the AAIA/NASA Conference on Intelligent Robots in Field, Factory, Service and Space*, March 1994.

[2] E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.* PhD thesis, Virginia Polytechnic Institute Department of Computer Science, April 1991.

[3] R. Gelin, J. M. Detriche, J. P. Lambert, and P. Malblanc. The Sprint of Coach. In *Proceedings of the '93 International Conference on Advanced Robotics*, November 1993.

[4] D. P. Miller. Rover navigation through behavior modification. In *Proceedings of the NASA Conference on Spacecraft Operations Automation and Robotics*, July 1990.

[5] M. G. Slack. Sequencing formally defined reactions for robotic activity: Integrating RAPS and GAPPS. In *Proceedings of the SPIE Conference on Sensor Fusion*, November 1992.

[6] S. Yu, M. G. Slack, and D. P. Miller. A streamlined software environment for situated skills. In *Proceedings of the AAIA/NASA Conference on Intelligent Robots in Field, Factory, Service and Space*, March 1994.

74

AIAA-94-1230-CP

DEXTERITY ENHANCEMENT IN MICROSURGERY USING TELEMICRO-ROBOTICS

Steve Charles, M. D.

Micro-Dexterity Systems, Inc.

6401 Poplar Avenue, Suite 296

Memphis, TN 38119

901/767-6662

S32-63

[illegible]

207076

p. 1

N94- 30579

The presentation will focus on finding the spectrum of dexterity performance while performing microsurgery in various specialties. It will be noted that individuals very markedly in their performance in the position, velocity, stability, and force domains. There are surgeons who have a tremor who otherwise move very slowly and carefully while there are other surgeons who apply excessive force, but never have a tremor or move excessively fast. There are yet other surgeons who move excessively fast, yet they do not have a tremor.

Dexterity enhancement includes position down scaling, tremor filtering, fatigue elimination, and other second-order issues such as confining the work space, velocities, accelerations, or forces.

It will be described that the hand's position performance is degraded when it is asked to actuate the tools and that remote actuation alone increases the positioning capabilities. It will be noted that rotary and telescopic functions are far more difficult than writing or engraving-like motions.

The safety issues concerning velocities and forces will be discussed and the need for impedance control pointed out. Simplistically, the devices should be made with variable compliance so that they can function rigidly as a robot would or compliantly as a human would, depended on the setting of this parameter.

Tool interfaces will be discussed with an emphasis on the overall performance of the position, end effector, and tool as a unit. Space constraints, force, and velocity requirements will be discussed in this section as well.

Referencing the coordinate system to pre- or inter-operative imaging systems will be discussed as well as an emphasis on the system architecture.

6075

AIAA-94-1232-CP

An Intelligent Robotic Aid System for Human Services

N94-30580

K. Kawamura, S. Bagchi, M. Iskarous, R. T. Pack, and A. Saad
Center for Intelligent Systems
Box 1804, Station B Vanderbilt University
Nashville, TN 37235

[illegible]

Abstract

The long term goal of our research at the Intelligent Robotic Laboratory at Vanderbilt University is to develop advanced intelligent robotic aid systems for human services. As a first step toward our goal, the current thrusts of our R&D are centered on the development of an intelligent robotic aid called the ISAC (Intelligent Soft Arm Control). In this paper, we describe the overall system architecture and current activities in intelligent control, adaptive/interactive control and task learning.

I Introduction

The goal of our current research is to develop an intelligent robotic aid system for the service sector such as hospitals and home. The main benefit of such a system is to provide the sick and physically challenged person with means to function more independently at home or work place. As a first step toward our goal, we have developed a prototype robotic aid system called the ISAC (Intelligent Soft Arm Control).[1] To insure ease of use, safety, and flexibility of the system, we have integrated several sensors such as vision, voice, touch and ultrasonic ranging. The user interacts with the system in natural language like commands such as *'feed me soup.'* Other related R&D activities being conducted include the development of an ISAC/HERO cooperative aid system with a HERO 2000 mobile robot to extend the system capabilities and work on a flexible microactuator robotic hand. In this paper, the overall system architecture is first presented. Next, the construction and performance of a parallel controller is described, followed by a discussion on various command interpreters and reflex control. Very preliminary results from recently constructed macro action builder and task learning module follow to illustrate the ease of use. We conclude with a discussion of the remaining technical issues needed to be addressed.

II System Architecture

ISAC is a robotic aid system for feeding the physically handicapped. It uses a unique manipulator called Soft Arm. The Soft Arm is a pneumatically-actuated manipulator. It is lightweight and suitable for operation in close proximity to humans. The actuators are fiber-reinforced rubber tubes called ***rubbertuators***, whose length depends on the pressure of the air inside the tube. Two rubbertuators control a joint in much the same way as human muscles.

The feeding task requires the recognition and the location of objects such as spoon, fork, and bowl on the table so that the arm can manipulate them. These objects are recognized from an image taken by an overhead camera. The recognition is independent of the size and orientation of the objects, a requirement characteristic of a normal feeding environment where utensils of different sizes are present at various orientations. ISAC also uses stereo cameras to track the face of the user in 3-D. This allows the arm to reach the mouth of the user even when he moves his head. Real-time face tracking also allows the detection of a sudden motion of the user. This could be caused by a sneeze or a muscle spasm. In such a case, the arm uses reflex action to move away from the path of the user.

Figure 1 illustrates the integrated hardware/software configuration of the ISAC system.[2] As shown in the figure, ISAC has a distributed architecture. The distributed architecture will allow us to easily add new modules and, therefore, new functionality.[3] The breakdown of one module will not halt the system, but rather, it will only result in a degradation of the activities that the system as a whole could previously perform.

The nature of communication between the modules must be such that each can exist without assuming the existence of other modules. This is achieved with a "blackboard," which is a means of indirect communications between modules.[4] Whenever a module requires a service to be performed by some other module, it posts the request to the blackboard. The requests in the blackboard are monitored by the modules, which perform the ones they are capable of. A brief description of key modules in ISAC are given below.

Object Recognition The object recognition module captures an image of the environment and identifies the location of all recognizable objects. The recognition algorithms used in this module is described in Bishay *et al.*[5] Recognition is model based, using a normalized distance histogram to find the best

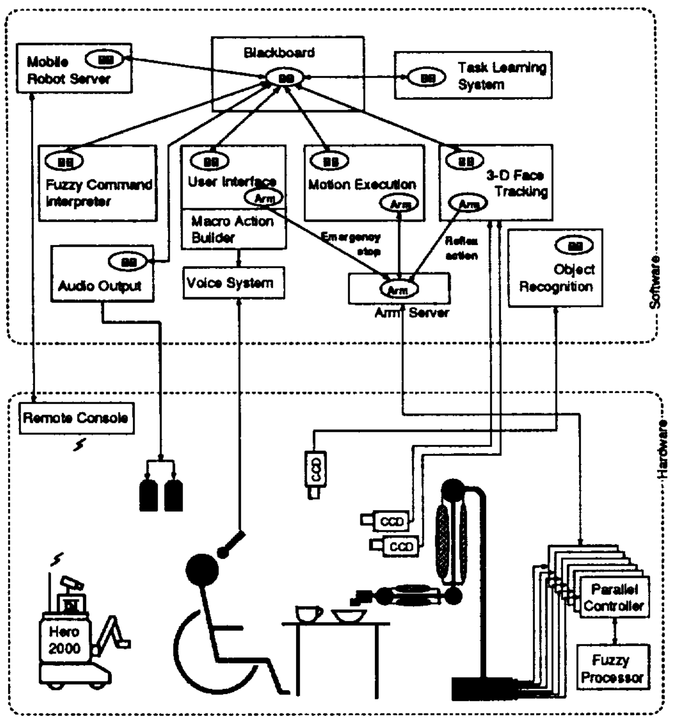


Fig. 1. Integrated ISAC architecture.

match with histograms in a database of possible target objects. An orientation histogram is used to determine the orientation of the object. Example objects in the environment are spoon, fork, knife, cup, and bowl. The recognition algorithm is robust enough to recognize various types and sizes of generic objects such as a spoon.

Face Tracking The face tracking module uses a pair of cameras to determine the position of the face in 3-D. The forehead of the user is tracked by both cameras. The disparity between the 2-D position obtained from each camera provides the third dimension: the distance of the user from the camera. Details of the tracking algorithm and camera calibration are described in Ernst *et al.*[6] In addition to specifying the position of the user's face for accurate feeding, the face tracking module can also detect any sudden motion made by the user. If the direction of the motion is towards the arm, such that a collision with the arm is possible, the arm is moved away from the user in a reflex action.

Voice Recognition A voice recognition system replaces the keyboard as the main user interface. Currently we are using the IN^3 commercial voice recognition system. It is running in parallel with the planning process, allowing the user to intervene the task execution if necessary.

Parallel Control The Soft Arm is controlled by a transputer-based parallel controller. It uses a network of transputers that can be reconfigured in case of a fault in the controller. Details of this module is given in Section III. We are developing a control system which can learn the best control strategy using a neural network and fuzzy logic. The neural network will be used to generate the knowledge base which will be used by the fuzzy controller.

Macro Action Builder This module acts as a voice-based "teach pendant" for the system designer or user. It provides the user with the ability to teach ISAC new actions and later retrieve them. It also allows the user to use fuzzy and context dependent commands such as *move closer*. This module enhances the extensibility of the ISAC's tasks as described in Section V.

Task Learning This module, currently under development, adds the capability of learning from obeying user commands and observing their effects. While the action building facility allows the system to learn *how* to perform an action, this module learns *when* and *why* to perform it, allowing the system to learn how to plan. The learning mechanism is described in Section V.

The ISAC system identifies some key requirements of service robot systems. These form the objectives of many research areas such as control, user interface, planning, and learning.[7,8] The following sections highlight these important research issues and describe the work being performed in greater detail.

III Intelligent Control

Parallel Controller

One of the key issues in the ISAC system is intelligent control. Since the Soft Arm exhibits a highly nonlinear joint dynamics due to rubbertuators[9], a special controller is needed to allow different control techniques to be used. Currently a transputer-based parallel controller is designed and implemented to control the Soft Arm as shown in Figure 2. It consists of a network of eight transputers. Each joint of the Soft Arm is connected to one transputer as its joint controller. A master transputer is then used to communicate with the host computer and supervise the joint controllers. The master node contains the robot command interpreter and the kinematic model of the Soft Arm.[10] It is also connected to the fuzzy processor FP-3000 which acts as a fuzzy coprocessor for control and path planning.

Currently, a PID controller is implemented in each joint node. A cubic spline is generated as the trajectory for the joint motion to follow. This reduces the amount of energy stored while moving the joint, thus allowing the speed to be increased without considerable jerky motion due to the nonlinearities of the rubbertuator joint.[11] The controller can access all the motion data and issue a new command at any time even when

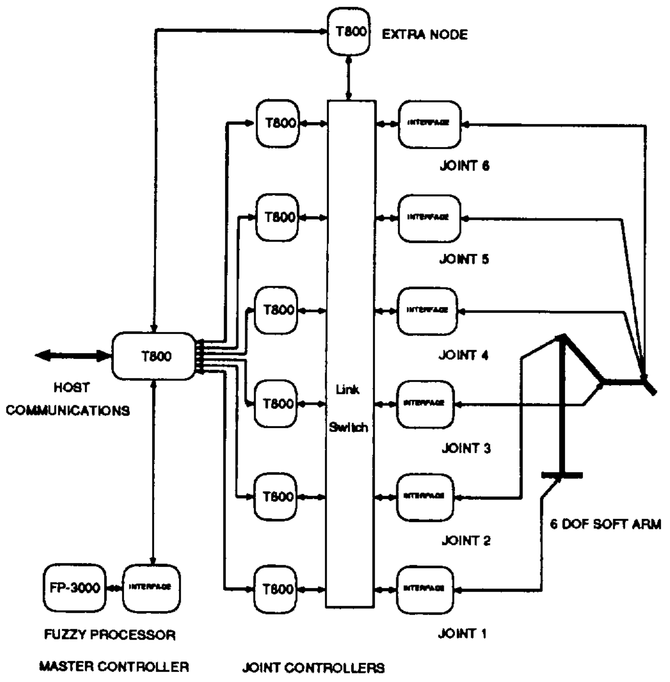


Fig. 2. The Transputer-based Parallel Controller.

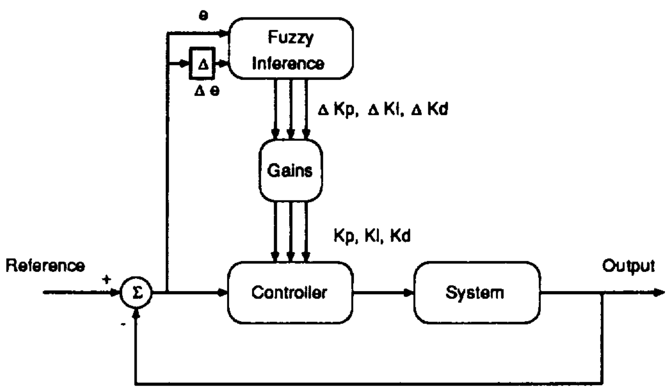


Fig. 3. Fuzzy Tuning for the Joint PID Controller.

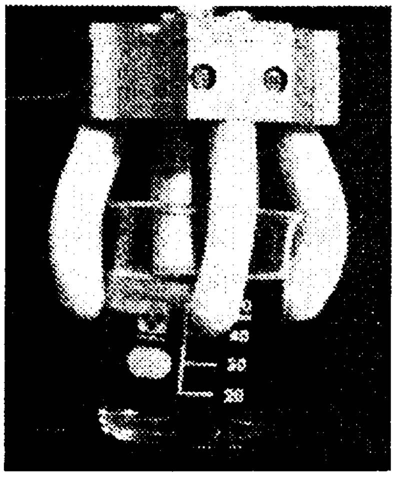


Fig. 4. The Flexible Microactuator.[14]

the robot is still moving. This is very important to implement the reflex action.

The nodes of the parallel controller are connected together via a programmable link switch. This feature allows the controller to be reconfigured in case of a fault detected in one of its nodes. A spare transputer can replace the faulty one by reconfiguring the connections of the network.

Fuzzy Control

Currently, a fuzzy tuning mechanism is used to tune the parameters of each PID controller as shown in Figure 3. This is useful with rubbertuators because of their nonlinear behavior. This mechanism uses three fuzzy matrices to tune the proportional, integral, and differential gains of the PID controller. Each matrix is similar to the Macvicar-Whelan matrix described in Tzafestas*et al.*[12] The fuzzy supervisor continually updates the controller parameters based on heuristic rules. The output of the fuzzy supervisor is the amount of change in each parameter of the PID controller. This allows the designer to specify different conflicting performance indices such as the trajectory following and disturbance rejection which leads to improved performance of the transient and steady state behavior of the closed loop system. In this case, the fuzzy system handles joint couplings as disturbances.

Combined with fuzzy logic, neural networks can also be used to learn the human-like trajectory to be followed by the robot. This technique uses neural networks to generate the knowledge base used by the fuzzy system.[13]

Flexible Gripper

Recently, research on the use of a flexible gripper has started. The gripper is composed of four flexible microactuators which act as fingers (shown in Figure 4). Each finger has three degrees of freedom — pitch, yaw and stretch.[14]

The flexible microactuator is made of fiber-enforced rubber. Internally, it is divided into three chambers whose individual pressures are controlled independently. The tip of each microactuator can be positioned depending on pressure differences inside its chambers.

These microactuators are very useful in applications that require flexible grippers. The flexible microactuator can also be very useful in zero gravity applications since it will not be loaded with object weight. The use of the flexible gripper will also increase the variety of tasks ISAC can perform such as handling fragile objects.

Issues on position sensing and closed loop control of the microactuator need to be investigated. Currently, open loop control is used to drive the flexible microactuator. To use closed loop control for the microactuator, force sensitive resistors will be used for position and force sensing.

C-6.

IV Interactive Control

The ISAC system's chief purpose is to interact with its user in a friendly and beneficial fashion. ISAC must provide a simple and consistent user interface with plenty of feedback so that the user is not intimidated or misunderstood. The system should be capable of handling the terms people use in normal language and it should understand when context applies to a command. At the same time, the system must keep a vigil for potentially dangerous situations and react to these without needing user interaction.

Command Interface

If ISAC is to be useful as a tool for a physically challenged person, the system must have a simple and flexible user interface. We elected to use voice commands to drive the system. The user's voice is captured by a microphone and is processed by a voice recognition system. Only a short training session is required to handle any speaker.

After the words are recognized, the planning module takes over and breaks down the user commands into actions for the system. Thus, the simple command *'feed me soup'* is broken down into picking up the spoon, going to the bowl, dipping into the bowl, tracking the user's face, and positioning the spoonful of soup at the user's mouth. These commands are also broken down into direct actions and coordinates for the Soft Arm. The current system handles natural-language-like commands by repeatedly breaking down the commands into subcommands until primitive actions are reached.

Another important aspect of the ISAC user interface is feedback. The ISAC system provides voice feedback by means of digitized messages that are replayed under certain conditions. These messages acknowledge user commands and assure the user that the system is doing what is expected, before things have gone too far. The messages also transmit error conditions to the user. At the moment ISAC detects a situation when it cannot pick up a spoon, even though the user requested soup, it will report an error to the user. Thus voice feedback is used to make ISAC more natural to use and to report error conditions in a straightforward manner.

Fuzzy Command Interpreter

A recent addition to the ISAC system is a fuzzy command interpreter. This module looks at user commands that contain fuzzy linguistic terms and translates them into crisp outputs for the rest of the ISAC system given the current context as shown in Figure 5. This context-based translation is a very powerful mechanism and allows a series of commands to be replaced by one fuzzy command (for example, *'move a lot closer'* would position the robot close to the user, and the subsequent command *'move closer'* would only move the robot a

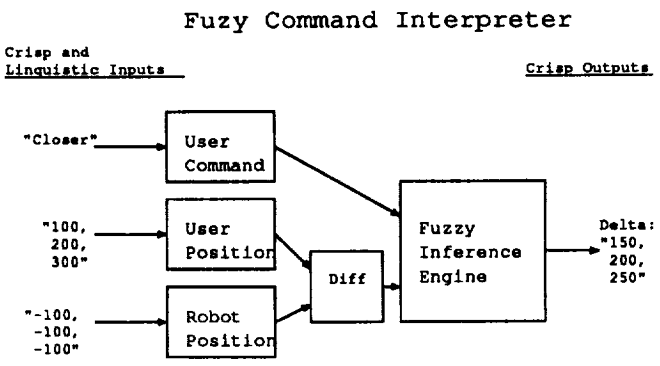


Fig. 5. Fuzzy Command Interpreter.

tiny bit because the arm is already "close" to the user). Fuzzy inference is used to take these linguistic terms and generate the crisp outputs which ISAC can use. This mechanism adapts user commands based on current system context information.

By having the ability to understand fuzzy linguistic terms in commands, ISAC's user interface is much friendlier to potential users and is also more powerful due to the fact that these commands include context as well as "fuzziness."

Reflex Action

One important characteristic of an intelligent robotic system is the ability to detect a potentially dangerous condition and react to this condition without user intervention. In particular, when a robot is in close proximity to people it is very important that the robot should not injure the person even in "emergency" situations. To this end ISAC is equipped with a reflex system like the one described by Kara *et al.*[1] The key system components related to reflex action are the stereo real-time face tracking system, the sonar sensor, and the parallel controller.

The reflex system monitors the user's position relative to the arm position and when the user makes a sudden motion toward the arm (as in a sneeze or convulsion), a high speed signal is sent to the arm controller to immediately move the robot out of the user's way. This type of intelligence is crucial in insuring that ISAC performs well under user command as well as situations that the user did not expect.

Stereo Face Tracking The ISAC system relies on stereo face tracking[6] to get the user's position. This tracking system locks on to the user and can track the user's position at 10-12 frames/sec. By using stereo cameras the face tracking system can track objects in 3 dimensional space and provides z (depth) values as well as x,y position. This depth value is the one that is crucial to the reflex action. The 3D face tracking system is shown in Figure 6.

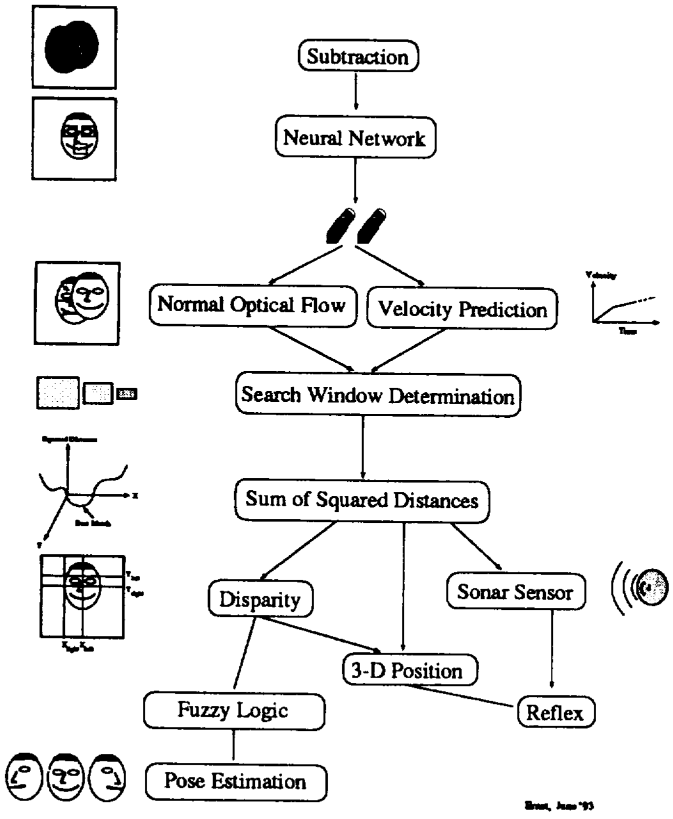


Fig. 6. 3D Face Tracking System.

Gripper Mounted Sonar Sensor In addition to the stereo face tracking, the ISAC system uses a sonar sensor mounted on the gripper to measure the relative distance from the robot to the user. This sensor provides additional information about the relative position and velocity of the user with respect to the arm. The sampling rate of the sonar sensor is slightly faster than the cameras in the face tracking system and thus provides better tracking information to control the reflex action.

Parallel Controller and Fuzzy Supervisor
The reflex action depends on the ability of the arm to move quickly out of the way. Ideally the servo system for the arm could complete any motion that we desire, but this is not the case. One type of control is useful for making smooth steady motions during normal system operation, but the control system response should be entirely different during the reflex action, where response speed is critical. Due to this need, our parallel controller[10] uses a fuzzy supervisor.[12] to tune the control loops as described in Section III. During normal operation, the fuzzy supervisor sets the controller gains for steady smooth motions, with low overshoot and high damping. When a motion command meets the requirements of reflex, the gains are set to minimize the rise time only. Thus, the damping ratio and other indices are ignored during the reflex motion. This results in quick, but jerky motions.

Reflex The two sensor systems allow the reflex system to keep constant watch over the possibility of user injury and the fuzzy supervisor in the controller tunes the arm for the best response in the emergency situation. The combination of these systems leads to an effective reflex action to protect the user from injury.

V Task Learning

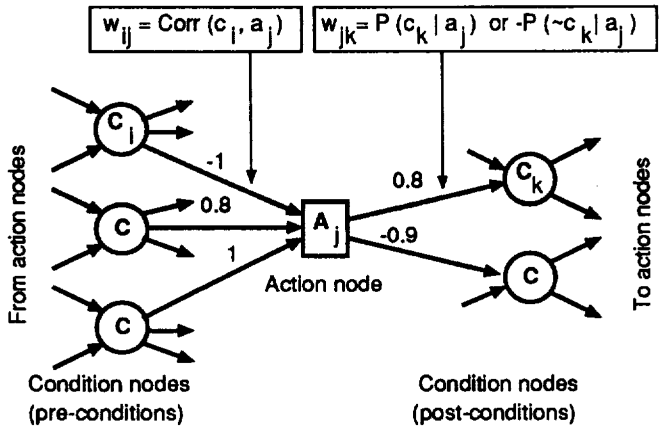
An aid system that comes.preprogrammed with a fixed repertoire of tasks will not be of help to users with unanticipated needs. The advantage of using a general purpose robot manipulator over the specific-purpose aid devices cannot be fully realized unless the user can create new tasks for the robot. Teleoperation has traditionally been the way by which the user can move the arm to perform the action desired by the user. However, users find teleoperation very tiring and prefer to substitute them with high level commands.[15] To achieve this, the system must be able to create high level actions out of teleoperated commands and then use a sequence of these actions to carry out tasks.

The knowledge to be learned can be divided into three types: how to perform an action, when to perform it, and what are its effects. To address the first type, we have designed an action builder which allows the user to create macro actions from primitive motion commands and existing macro actions. This process is described in the next subsection. For the system to plan it must learn the two remaining knowledge types which represent the preconditions and effects of an action. These can be learned when the user prompts the robot to perform an action. The conditions existing in the environment just before the action was performed and the conditions changed as a result of the action are used to induce the preconditions and effects. The learning algorithm is described later in this section.

Figure 7 shows the knowledge representation used for planning. Actions and conditions are represented as nodes and the relation between them as links. Learning the relations between actions and conditions involves formation of these links. Planning occurs through a spreading activation process: "Potential" from the goal conditions are spread backwards to the actions that can achieve them. Similarly, potential from the current state of the conditions is spread forward to actions. An action is performed when its potential rises above a predefined threshold value. Details of the task planning and learning mechanisms are described in Bagchi *et al.*[16]

Action Builder

Traditionally, the user of ISAC was supposed to rely on the knowledge precompiled by the system developer for



w_{ij} = Corr (c_i, a_j)
w_{jk} = P (c_k | a_j) or -P (~c_k | a_j)
From action nodes
-1
0.8
1
0.8
-0.9
Action node
To action nodes
Condition nodes (pre-conditions)
Condition nodes (post-conditions)

Fig. 7. Representation of an action, its preconditions, and effects.

the execution of a high level command such as *'feed me soup'*.[2] The objective of the Action Builder module is to enable the user to increase the system's repertoire of actions, by teaching new actions and integrating them with those already provided to the user.

Any complex robotic action is made of a sequence of several primitive actions. Primitive actions are those that can be directly executed by the robot. In our case, they fall under two categories:

- *unconditional* motions which instruct the Soft Arm to get the tip of the gripper to a certain location within its workspace, such as 'move to location xyz,' and
- *conditional* motions which are tied to the input from any of the sensors mounted on the Soft Arm, such as 'move down until an input from the photocell sensor is detected.'

The role of the system developer is then to provide the user with an initial set of complex actions, and with a library of primitive actions. The primitive actions incorporated into such a library enable the user to exploit the Soft Arm, as well as any of the other modules that ISAC comprises. The user then can tailor a complex action by combining any number of primitive and/or complex actions. A user-defined complex action could, in turn, be at the basis for creating actions of higher complexity. Hence, this module ensures the extensibility of ISAC's repertoire of actions in order to accommodate the specific needs of its user.

User Interface   The user interface provided by the the Action Builder module is highly user-friendly. From the user's perspective, the Action Builder module acts as a voice-activated "teach pendant." It accepts the user-defined complex actions, stores them, and retrieves them whenever the user deems it necessary. Once retrieved, the user can modify or delete any of the previously stored complex actions or alternatively use them, in conjunction with the primitive and complex actions provided by the system developer, in order to build a more complex action.

Learning from Observation

The learning task can be divided into two related parts: learning the effects of an action and learning its preconditions. For both, learning is supervised. The user asks the robot to perform a set of actions. As the robot performs them, it observes the change of conditions in the environment and induces relations between conditions and actions. The system can plan for a task once the correct relations have been learned. It should be noted here that the system does not learn the sequence of actions that can achieve the goal. Instead, it learns how to make the procedural actions "transparent," by associating preconditions and effects with each of them.

Learning the Effects of an Action   The effects of an action can be easily identified if they can be detected as soon as the action is complete. For robotic tasks where objects have to be manipulated, this requirement is true. The task of the learning system is not only to detect the conditions that change after an action is performed, but also to maintain a probability for this change. The strength of the connection between an action a_j and a condition c_k (see Figure 7) is given by

$$w_{jk} = \begin{cases} P(c_k \mid a_j) = \text{num}(c_k, a_j)/\text{num}(a_j) \\ \qquad\qquad \text{if } c_k \text{ changes to true} \\ -P(\neg c_k \mid a_j) = -\text{num}(\neg c_k, a_j)/\text{num}(a_j) \\ \qquad\qquad \text{if } c_k \text{ changes to false} \end{cases} \tag{1}$$

where $\text{num}(c_k, a_j)$ is the number of times c_k is true after action a_j was performed and $\text{num}(\neg c_k, a_j)$ is the number of times it is false. $\text{num}(a_j)$ is the number of times the action was performed.

It is possible for this approach to identify spurious conditions as effects. Conditions can change at random or as a result of other agents in the environment. However, as the action is performed a number of times, the strength of the link to a uncorrelated effect will decrease.

Learning the Preconditions of an Action
Preconditions define the situations under which an action will be successful in enabling all its effects. It is difficult to discover the preconditions because one cannot be sure that an action failed because of incorrect preconditions or because of the unreliability of the environment. When the robot is asked by the user to perform an action, the entire state of the environment may be assumed to be the precondition. This, however, is too specialized: the learning mechanism must generalize the precondition set over multiple instances of successful operation of the action.

This generalization is performed by maintaining correlation statistics between the state (true/false) of the conditions and the successful execution of an action. The correlation measure used is given by

$$w_{ij} = \text{Corr}\,(c_i, a_j) = P(a_j \mid c_i) - P(a_j \mid \neg c_i), \qquad (2)$$

where $P(a_j \mid c_i)$ is the probability of action a_j succeeding given c_i is true, and $P(a_j \mid \neg c_i)$ is the probability of action a_j succeeding given c_i is false. These probabilities are approximated from the statistics kept from multiple executions of the action. For "hard" preconditions, those that *must* be in the desired state for the action to succeed, the correlation will be 1 (if the condition must be true) or -1 (if the condition must be false). When the value is between these extremes, the precondition is termed "soft," denoting desirability but not necessity. Finally, a value of 0 (or close to it) denotes no correlation between the action and the condition.

Examples

Equipped with an initial set of primitive actions, the user can create a complex action to pickup a fork by using the following steps:

1. *'locate objects'* to locate the objects on the table, using the object recognition module.
2. *'goto fork'* instructs the Soft Arm to move the tip of the gripper on top of the fork's location.
3. *'move down'* until an input from the photocell sensor is detected.
4. *'close gripper'* to grasp the fork (now positioned between the gripper's fingers).

These steps are then stored as the *'pickup fork'* complex action.

Once, the system has been taught how to perform the action *'pickup fork'* it has to learn its preconditions and effects. Consider the following state of the conditions when the user asked the action to be executed for the first time (the ~ symbol denotes false and its absence denotes true):

```
located (spoon),
located (fork),
holding (nothing),
~in (soup, bowl),
in (plate, fries),
...
```

These states form the initial preconditions for the action. After *'pickup fork'* is executed, the conditions that changed are observed. For this example, the state of the changed conditions are:

```
holding (fork),
~holding (nothing).
```

These form the initial effects.

At this stage, the preconditions are too specialized. For example, `located (spoon)` should not affect the outcome of the action in any way. The preconditions are generalized from repeated observations. For example, if `located (spoon)` is false when *'pickup fork'* is performed for the second time, the correlation is changed to zero. After a series of such observations, it is expected that all uncorrelated conditions will have low link strengths and the preconditions will converge to:

```
located (fork),
holding(nothing).
```

The effects are also updated every time the action is performed. This allows the maintenance of statistics that allow the determination of the probability of an effect occurring when the action is performed. This information is used by the planner when it has to choose between multiple action sequences in order to achieve its goals with the highest reliability.

VI Conclusions and Future Directions

We have presented the design and implementation of an intelligent robotic aid system for human services. The prototype system, termed the ISAC (Intelligent Soft Arm Control) has been shown to be an excellent testbed for such a system which may be used in the service sector in the future. Figure 8 shows ISAC in its current working environment. Remaining technical issues to be addressed include the development of intelligent control mechanisms for the flexible microactuator, integration of the learning algorithm with the ISAC system and the development of a robust real-time sensor fusion algorithm for the ISAC/HERO system.

VII Acknowledgments

The authors would like to thank the members of the Intelligent Robotics Laboratoty for their contributions to the ISAC project. This research is partially supported by Vanderbilt University, Bridgestone Corporation and Toshiba Corporation.

References

[1] A. Kara, K. Kawamura, S. Bagchi, and M. El-Gamal, "Reflex control of a robotic aid system for the physically disabled," *IEEE Control Systems Magazine*, vol. 12, pp. 71–77, June 1992.

[2] S. Bagchi and K. Kawamura, "An architecture of a distributed object-oriented robotic system," in *IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS '92)*, (Raleigh, NC), pp. 711–716, July 1992.

[3] H. Asama, M. K. Habib, I. Endo, K. Ozaki, A. Matsumoto, and Y. Ishida, "Functional distribution among multiple mobile robots in an autonomous and decentralized robot system," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, vol. 3, (Sacra-

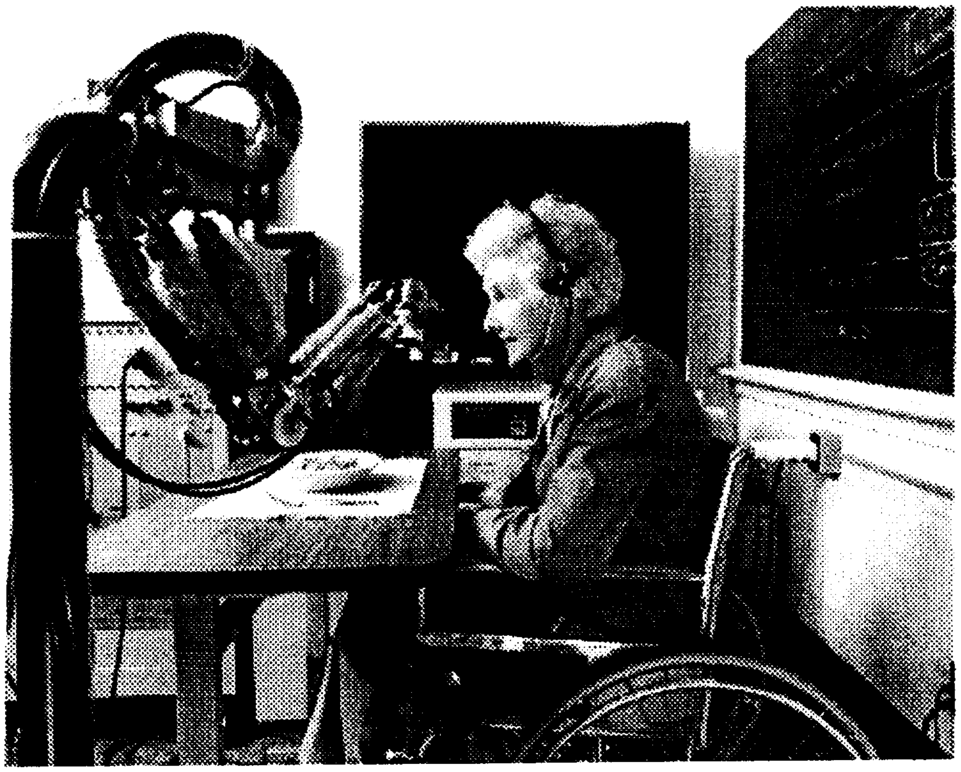


Fig. 8. ISAC at work.

mento, CA), pp. 1921–1926, IEEE Computer Society Press, 1991.

[4] R. Engelmore and T. Morgan, eds., *Blackboard systems.* Reading, MA: Addison–Wesley, 1988.

[5] M. Bishay, K. Homma, C. Swain, K. Fujiwara, and K. Kawamura, "Two-dimensional object recognition," Tech. Rep. CIS-93-01, Center for Intelligent Systems, Vanderbilt University, Nashville, TN, 1993.

[6] R. Ernst, M. El-Gamal, R. A. Peters II, and K. Kawamura, "3-D face tracking in ISAC," Tech. Rep. CIS-93-06, Center for Intelligent Systems, Vanderbilt University, Nashville, TN, 1993.

[7] J. F. Engelberger, *Robotics in Service.* Cambridge, MA: The MIT Press, 1989.

[8] K. Kawamura and M. Fashoro, "Biorobotics," Tech. Rep. CIS-92-05, Center for Intelligent Systems, Vanderbilt University, Aug. 1992.

[9] Bridgestone Corporation, *Rubbertuators and Applications for Robotics, Technical Guide No. 1*, 1986.

[10] M. Iskarous and K. Kawamura, "A fault-tolerant transputer-based parallel controller for the soft arm robot," *Proc. of the 6th Conference of the North American Transputer User Group (NATUG6)*, pp. 234 - 246, May 1993.

[11] M. Iskarous, R. T. Pack, and K. Kawamura, "A reconfigurable transputer-based parallel controller," *Submitted to the 1994 American Control Conference*, 1994.

[12] S. Tzafestas and N. P. Papanikolopoulos, "Incremental fuzzy expert pid control," *IEEE Transactions on Industrial Electronics*, vol. 37, pp. 365–371, Oct. 1990.

[13] B. Kosko, *Neural Networks and Fuzzy Systems.* Prentice Hall, 1992.

[14] K. Suzumori, S. Iikura, and H. Tanaka, "Applying a flexible microactuator to robotic mechanisms," *IEEE Control Systems*, pp. 21–27, Feb. 1992.

[15] M. A. Regalbuto, T. A. Krouskop, and J. B. Cheatham, "Toward a practical mobile robotic aid system for people with severe physical disabilities," *Journal of Rehabilitation Research and Development*, vol. 29, no. 1, pp. 19–26, 1992.

[16] S. Bagchi, G. Biswas, and K. Kawamura, "A spreading activation mechanism for decision-theoretic planning," in *AAAI Spring Symposium on Decision-Theoretic Planning*, Mar. 1994. In press.

[illegible]

AIAA-94-1233-CP

N94-30581

AN INTELLIGENT ROBOT FOR HELPING ASTRONAUTS

J. D. Erickson*, K. A. Grimm†, and T. W. Pendleton‡
National Aeronautics and Space Administration Lyndon B. Johnson Space Center
Houston, Texas 77058

[illegible]

p. 13

Abstract

This paper describes the development status of a prototype supervised intelligent robot for space application for purposes of (1) helping the crew of a spacecraft such as the Space Station with various tasks, such as holding objects and retrieving/replacing tools and other objects from/ into storage, and (2) for purposes of retrieving detached objects, such as equipment or crew, that have become separated from their spacecraft. In addition to this set of tasks in this low-Earth-orbiting spacecraft environment, it is argued that certain aspects of the technology can be viewed as generic in approach, thereby offering insight into intelligent robots for other tasks and environments.

Candidate software architectures and their key technical issues which enable real work in real environments to be accomplished safely and robustly are addressed. Results of computer simulations of grasping floating objects are presented.

Also described are characterization results on the usable reduced gravity environment in an aircraft flying parabolas (to simulate weightlessness) and results on hardware performance there. These results show it is feasible to use that environment for evaluative testing of dexterous grasping based on real-time vision of freely rotating and translating objects.

* Chief Scientist, Automation and Robotics Division, Member AIAA

† EVAHR Project Manager

‡ Head, Robotic Intelligence Section

1. Introduction

Numerous facets contribute to achieving robotic intelligence. This paper, based on a more complete presentation in reference 1, describes many of these facets and attempts to relate them to the central theme of a software architecture that enables a sufficient level of robotic intelligence and, thus, real work in real environments under supervision by exception. Related work by others is also outlined in reference 1. The essence of intelligent systems is that they are capable of collecting and applying knowledge of the situation gained at execution time and correlating it with other knowledge to take effective actions in achieving goals. Intelligent systems are composed of sensors for perceiving both the external and internal environments, effectors for acting on the world, and computer hardware and software systems for providing an intelligent connection between the sensors and effectors. Part of the processing by these computer systems is symbolic in a nonnumeric sense and thus enables practical reasoning, or the behavior which we humans call intelligent. The intelligent system we will be addressing, the Extravehicular Activity Helper/ Retriever (EVAHR), is a supervised, intelligent, mobile robot with arms and end effectors (see Figure 1). Intelligent robots of this nature are required for long-term operations in space and are mandatory for space exploration to improve safety, reliability, and productivity while enabling large cost savings through minimizing logistics[2].

Long-term space operations such as the Space Station have requirements for capabilities for rescue of extravehicular activity (EVA) crew and retrieval of equipment. A space station cannot chase separated crew or equipment, and other vehicles such as the Space Shuttle will not usually be available. In addition to the retrieval of drifting objects, another need is for robotic help to EVA crewmembers in various tasks, such as holding objects; retrieving and replacing tools and other items from and into storage; performing inspections; setting up and dismantling work sites; performing servicing, maintenance, and repairs; and deploying and retrieving payloads. Modeling, simulation, and analysis studies of space exploration missions have shown that supervised

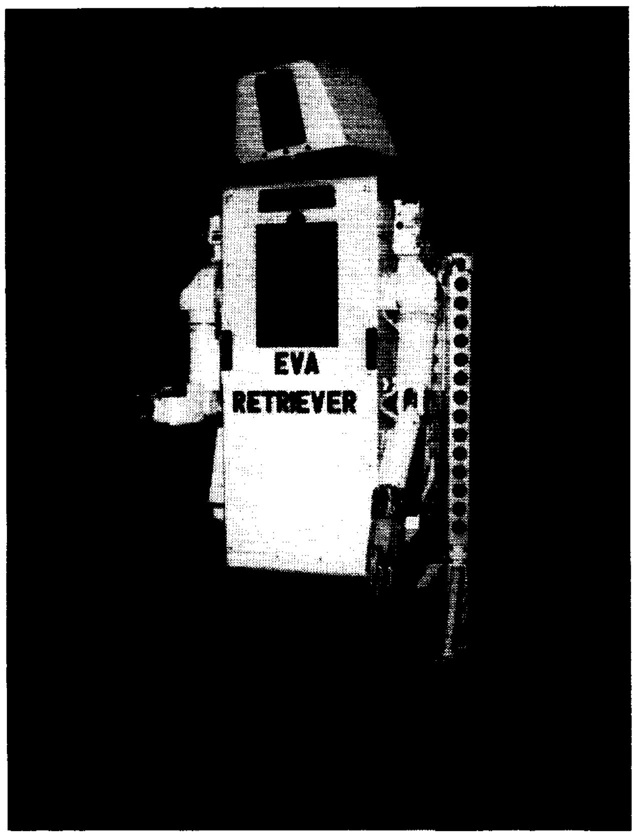


Figure 1. Phase II Retriever.

intelligent robots are enabling for human exploration missions[3,4].

The U.S. economy can reap major benefits from the development of supervised intelligent autonomous robotic systems[5,6], for such systems foster productivity improvements that raise the standard of living for everyone[7]. The solutions to the problems we will be solving to make the exploration of our solar system possible and practical will apply to the many critical problems we have on Earth which require operating in hazardous environments and to improving human productivity in many fields.

The free-flying, supervised intelligent robot called EVAHR is being prototyped as a potential solution to the crew helper and detached crew and equipment retrieval need. EVAHR is a technology test-bed providing evaluation and demonstration of the technology included for the following three purposes:

1. Robotic retrieval of objects which become detached from their spacecraft; e.g., astronauts adrift from the Space Station.
2. A robotic crew helper around a spacecraft; e.g., inspector, "go-fer," holder, maintainer, servicer, tester, etc.
3. A "generic" prototype supervised, intelligent autonomous robot (for planetary surfaces with different mobility such as wheels or tracks and for terrestrial applications with appropriate adaptations).

Early supervised intelligent robotic systems with initial capabilities to meet real needs are beginning to emerge from laboratories and manufacturers. It is now possible, in our opinion, to construct robots capable of accomplishing several specific high-level tasks in unstructured real-world environments.

The ability to acquire and apply knowledge and skills to achieve stated goals in the face of variations, difficulties, and complexities imposed by a dynamic environment with significant unpredictability is our working definition of "robotic intelligence." This does not require a broad-based general intelligence or common sense by the robot. However, doing the work needed to accomplish goals does require, in general, both mobility and manipulation in addition to reacting, or deciding "intelligently," at each step what to do. Further, supervised intelligent robots are required for human-robot teams where supervision is most naturally provided by voice.

Controlling supervised intelligent robots having both mobility and dexterous manipulation is a challenge[1], as is integration of sensing and perception into planning and control in a robust way.

Certain aspects of the EVAHR technology, which provide the capability for performing specified tasks in a low-Earth-orbiting spacecraft environment, can be viewed as generic in approach, thereby offering insight into intelligent robots for other tasks and environments. This is because the design of the software architecture, which is the framework (functional decomposition) that integrates the separate functional modules into a coherent system, is dictated in large measure by the tasks and nature of the environment. And because both the goal-achieving tasks and the partially unpredictable nature of the environments are similar on Earth and in space, the software architecture can be viewed as generic – as can many of the software modules, such as the AI planner, world model, and natural language interface. Other software is bundled with certain hardware. This leads to the concept of a modular, end-user customized robot put together from modules with standard interfaces[8-10] such as users do with a personal computer, yet maintaining real-time response.

2. Approach

The end goal for intelligent space robot development is one or more operational robots as part of human/robot teams in space. Prior to that, an evaluation of performance in space will be required.

Our approach to development of operational robots as part of human-robot teams in space is a systems engineering approach with iterative, three-ground-phase requirements prototype development, tested in both ground and aircraft simulations of space, followed by evaluation testing of a flight test article in space. We adapt and integrate existing technology solutions.

The EVAHR ground-based technology demonstration was established to design, develop, and evaluate an integrated robotic hardware/software system which supports design studies of a space-borne crew rescue/equipment retrieval and crew helper capability. Goals for three phases were established. The Phase I goals were to design, build, and test a retriever system test-bed by demonstrating supervised retrieval of a fixed target. Phase II goals were to enhance the test-bed subsystems with significant intelligent capability by demonstrating arbitrarily-oriented target retrieval while avoiding fixed obstacles. Table 1 summarizes some of the characteristics of the Phase II system. The objectives for Phase III, which is currently in progress, are to more fully achieve supervised, intelligent, autonomous behavior by demonstrating grasp of a moving target while avoiding moving obstacles and demonstrating crew helper tasks. Phase III is divided into two parts. Phase IIIA goals are to achieve real-time complex perception and manipulator/hand control sufficient to grasp moving objects, which is a basic skill both in space retrieval and in accomplishing the transition from flying to attaching to a spacecraft. Phase IIIB goals are to achieve a software architecture for manipulation and mobility, with integrated sensing, perception, planning, and reacting, which guarantees safe, robust conduct of multiple tasks in an integrated package while successfully dealing with a dynamic environment.

Our overall testing approach is short cycle run-break-fix[11] with increasing integration and more relevant environments; such an approach finds design and implementation problems early when they are lowest cost to fix.

3. Hardware Design

The performance characteristics of the EVAHR hardware enable (or defeat) the "intelligent" behavior of the robot as "animated" by the software. We are testing only a subset of the Phase IIIB hardware in Phase IIIA.

The hardware subset includes a 7-degree of freedom (DOF) arm (Robotics Research K807i); a 5-DOF, compliant, force-limited dexterous hand; a laser range imager (Perceptron); a stereo video camera system (Teleos Prism 3); a pan/tilt unit; a 700 Megaflop computational engine employing Intel i860s and transputers; and an Inertial Measurement Unit (IMU) of accelerometers and gyros.

4. Software Design

During Phase IIIA we are using a subset of the reaction plan architecture while we are exploring two new approaches to the software architecture for Phase IIIB. The first is a version of the three-tiered, asynchronous, heterogeneous architecture for mobile robots[12-14] adapted to include manipulation. The second is a version of the SOAR architecture[15] applied to robots[16]. SOAR is of interest because of its capabilities in learning, including recent work in situated, interactive natural language instruction[17]. To be practical, the robot "programming" bottleneck must be avoided by using learning from experience and instruction to acquire skills and knowledge. SOAR has also been used to achieve resource-dependent behavior[18] and to learn reactive, stimulus-response rules, in addition to search control.

For each approach we are conducting evaluation testing of minimal prototype architecture implementations to obtain some evidence of their strengths and weaknesses for our tasks before selecting one for larger scale implementation in Phase IIIB. We present our evaluation results on SOAR in the section on results. We are not far enough along on prototyping the three-tiered architecture to have results yet.

Table 1. Unique and Special Aspects of Phase II EVAHR.

- Prototype supervised, intelligent, autonomous robot
- Voice commands provide goals and directions
- Clips into space-worthy Manned Maneuvering Unit (MMU) which has flown from Shuttle
- "Flies" by propelling pressurized gas from MMU thrusters it controls
- Self-locating in analogy to space use of global positioning satellites where retriever uses camera, gyroscopes, and accelerometers
- Builds its own internal dynamic knowledge of its environment based on continuous sensory perception – No preprogrammed environmental model to which the environment must conform
- Planning/replanning based on goals and internal dynamic knowledge of its environment and constraints such as flight rules
  - Path planner for obstacle avoidance and rendezvous can reason in advance about the success of the mission
  - Actions are synchronized to events in the world through sensing of preconditions of planned actions
  - Deals with unpredictability by detection/replanning if needed
- Range image obstacle location and target tracking, orientation, and grasp location
- Acts to acquire knowledge about obscured target
- Maneuvers to optimize grasp success relative to target orientation
- Chooses between one-handed grasp and two-armed grapple, depending on target size it perceives
- Uses dexterous grasping with proximity sensors, compliant grasp, and force-limited grasp
  - Right hand has 5 proximity sensors
  - Left hand has 3 proximity sensors and 9 tactile sensors (3 per finger)
- Uses pressure sensors on chest for two-armed grapple of large targets
- Uses fourteen 10-MIPS transputers, six 68020 controllers, and one 80386 processor in a hierarchical, distributed architecture

Safety is a major issue in human-robot teams, especially in space. Since robotic motion control programs cannot be considered safe unless they run in hard real time, an approach which addresses this issue in a different manner from that of the three-tiered architecture is needed for comparative evaluation. We are pursuing the development of one such approach[19].

The following discussion is due to Schoppers[20]. A statement of the pivotal problem in successfully coupling symbolic reasoning with the ability to guarantee production of a timely response has recently been made: "The timing of actions taken by a real-time system must have low variances, so that the effects of those actions on unfolding processes can be predicted with sufficient accuracy. But intelligent software reserves the option of extended searching, which has very high variance"[21].

The AI community has responded to this dilemma in roughly three ways[22]. When building a system that must act in real time as well as reasoning, one can choose to

1. Subject the AI component of the system to hard deadlines. This effectively embeds the AI reasoner within the real-time system, and under time pressure, results in loss of intelligent function.

2. Refuse to subject the AI component of the system to hard deadlines, and have the real-time subsystem "do its best" with whatever commands the AI subsystem can generate in time. This effectively embeds the real-time subsystem within the AI system, and under time pressure, results in loss of timely control.

3. Refuse to subject the AI component of the system to hard deadlines, but let the AI components "negotiate" with the real-time subsystem to obtain a feasible schedule for task execution. This does not embed either subsystem within the other, and with proper selection of the real-time executive's task schedule, has the promise of remaining functional under time pressure.

The three-tiered approach is a category three approach, whereas we interpret SOAR to be a category two approach.

We can now summarize the state of the art. Simple control systems can get away with seeming to be "fast enough," but that approach becomes potentially very dangerous in more complex systems, particularly in intelligent systems where the set of tasks being executed changes over time. In a system that may perform any subset of N possible tasks, there are 2^N possible combinations of tasks, and it becomes impossible to test the performance of each combination by hand when N is large. Therefore, it becomes imperative to have automated support for obtaining a guarantee that the system can always perform in hard real time.

4.1 Three-Tiered Software Architecture

Combining all prior knowledge and knowledge sensed during a task requires that planning in advance can only be guidance, with control decisions as to what to do postponed until such time as the situation is being sensed and the task is being executed. This is the essence of Agre and Chapman's theory of plans-as-advice[23], and is a design principle underlying the three-tiered approach.

Several researchers[12-14] have developed the three-tiered architecture to enable faster, more efficient interaction with the world and to allow the planner sufficient time to make intelligent decisions. Decisions based on the details of the local world are postponed and a "sketchy" plan is passed on to the next layer. The three layers are the planner, the sequencer, and the reactive controller.

The responsibility of the planning layer is to determine which tasks would accomplish the goal and in what approximate order. Thus, the planning layer forms a partially ordered set of tasks for the robot to perform, with temporal constraints. This plan is somewhat sketchy since not every detail of implementation, which would be determined by the current situation, is included. The AI planner which we are evaluating for this application is the AP Planner[24]. It may be possible to use SOAR for this application.

The sequencing "middle" layer is responsible for controlling sequences of primitive physical activities and deliberative computations. Operating asynchronously from the planner, yet receiving inputs from that layer, the sequencer takes the sketchy plan and expands it based on the current situation. Thus, the hierarchical plan expansion happens at execution time rather than at the deliberative stage. To implement the sequencer, data structures called Reactive Action Packages (RAP's) are used to represent tasks and their methods for executing[13].

At the lowest level, the reactive controller accepts sensing data and action commands, sensorimotor actions that cannot be decomposed any further, from the sequencer. For example, "move," "turn," or "grasp" are all examples of action commands that are passed onto the hardware. The reactive controller also monitors for success or failure of these commanded activities.

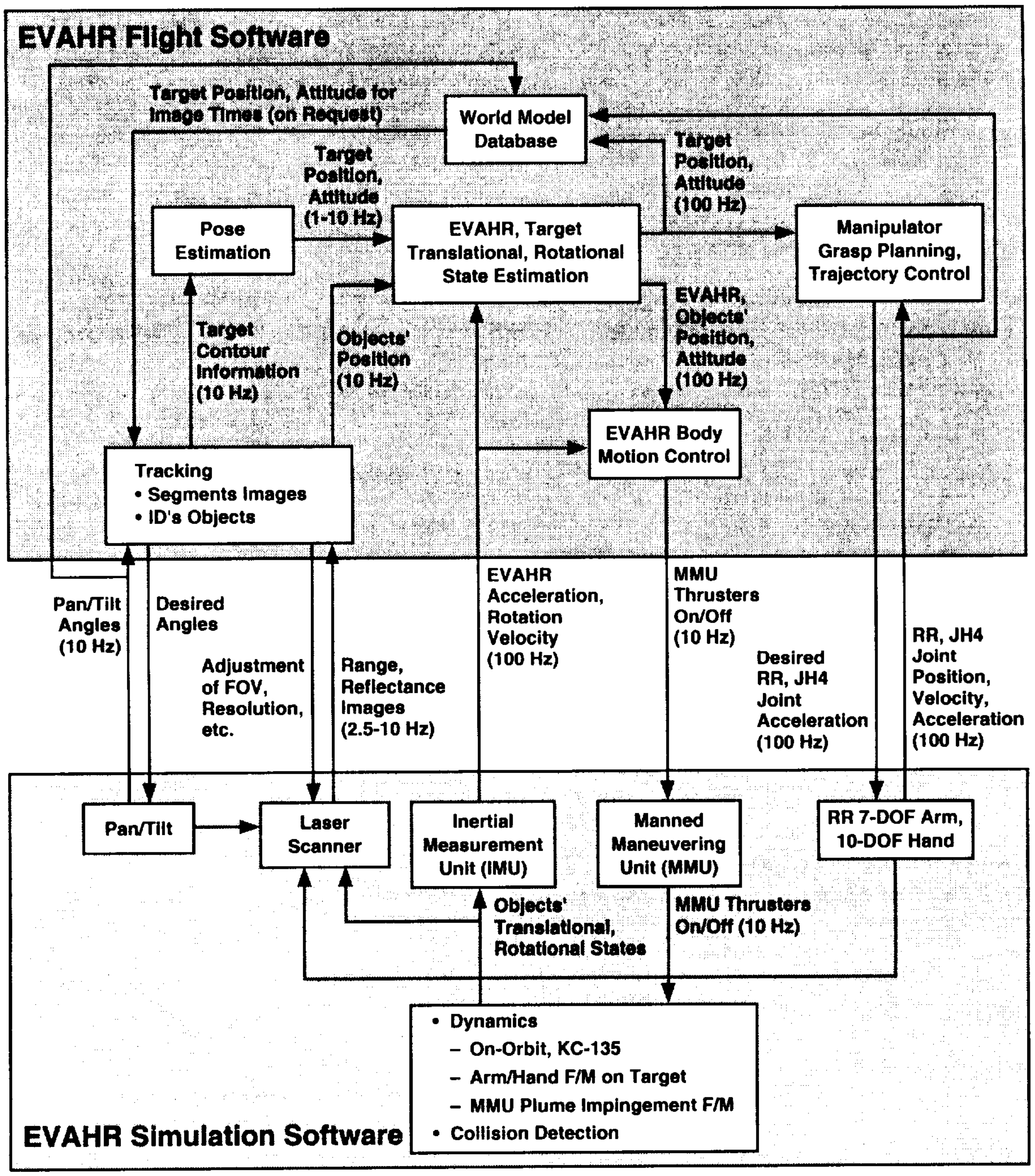
4.2 Phases IIIA and IIIB Software Architecture

The EVAHR Phase IIIA software is composed of sensing, perception, world modeling, planning, and acting. Figure 2 shows the relationship among these elements for the on-orbit retrieval problem where a free-floating target must be rendezvoused with, grasped, and returned. As tasks are added to the crew helper's repertoire in Phase IIIB, additional elements must be added to support AI planning, force feedback arm control, and voice interaction with the crew.

Sensing software provides the low-level interface to the hardware sensors, reading and time tagging sensor data and providing preprocessing to account for the effects of nonideal sensors. Sensing software also provides an interface to perception.

Visual sensing software is the primary module for acquiring information about the environment via optical sensors such as the 10 image/sec laser scanner and the 30 dual-image/sec stereo vision system. Software for voice and data reception (Phase IIIB) handles speech recognition, Global Positioning System (GPS) decoding, and design and operations knowledge support system (DOKSS) interfacing. Software for force/torque sensing, proximity sensing, and tactile sensing provides data acquisition and time tagging.

Our proprioceptive sensing software reads and time tags the IMU accelerometers and gyroscopes, GPS, position sensors on the manipulators and hands, thruster firing sensors, position sensors on the pan/tilt unit, fault sensors throughout the hardware, and robot resource status sensors.



EVAHR Flight Software
Target Position, Attitude for Image Times (on Request)
World Model Database
Target Position, Attitude (1-10 Hz)
Target Position, Attitude (100 Hz)
Pose Estimation
EVAHR, Target Translational, Rotational State Estimation
Manipulator Grasp Planning, Trajectory Control
Target Contour Information (10 Hz)
Objects' Position (10 Hz)
EVAHR, Objects' Position, Attitude (100 Hz)
EVAHR Body Motion Control
Tracking
• Segments Images
• ID's Objects
Pan/Tilt Angles (10 Hz)
Desired Angles
Adjustment of FOV, Resolution, etc.
Range, Reflectance Images (2.5-10 Hz)
EVAHR Acceleration, Rotation Velocity (100 Hz)
MMU Thrusters On/Off (10 Hz)
Desired RR, JH4 Joint Acceleration (100 Hz)
RR, JH4 Joint Position, Velocity, Acceleration (100 Hz)
Pan/Tilt
Laser Scanner
Inertial Measurement Unit (IMU)
Manned Maneuvering Unit (MMU)
RR 7-DOF Arm, 10-DOF Hand
Objects' Translational, Rotational States
MMU Thrusters On/Off (10 Hz)
• Dynamics
– On-Orbit, KC-135
– Arm/Hand F/M on Target
– MMU Plume Impingement F/M
• Collision Detection
EVAHR Simulation Software

Figure 2. EVAHR Phase IIIA Flight/Simulation Software Architecture.

Perception software extracts understanding of the environment from preprocessed sensor and voice recognition receiver data.

Visual perception is carried out through a combination of various visual functions. Visual functions that have been implemented in software include search, tracking, and pose estimation. Other visual functions, such as those for object recognition, will be integrated in the near future. Pose estimation is calculating the orientation of an observed object in a given image. Our approach to pose estimation is known as image-based (or multiview based) pose estimation[25].

Natural language understanding processing (Phase IIIB) starts with a symbolic representation that Retriever can interpret and act upon, returning an appropriate response. Such systems are practical when limited to a specific domain and a well-defined application.

In general, our world model stores internal state representations of the external world at a high level of abstraction, which allows the implicit predictions associated with the state (that it will remain valid for some time) to more likely remain valid for the lifetime of the internal state[26]. For moving objects, however, we use world model state estimators to bring the past measurements of motion descriptors up to the present time.

Planning enables the EVAHR to take a high-level goal and decide which subtasks must be accomplished to move the system to the goal. This selection and ordering of subtasks becomes very challenging, particularly if the system is monitoring the consequences of actions, replanning, or juggling multiple goals with changing priorities.

The vision system planner has been described previously[27].

A mobility planner is responsible for determining an optimal positional and rotational trajectory for the robot's body. "Optimal" usually implies (1) obstacle avoidance between the points of departure and arrival and (2) minimization of time, distance, and/or fuel consumption. In orbital scenarios (e.g., Space Station) fuel is at a premium, although with astronaut rescue, time is more critical. For this purpose, a trajectory planner/controller was developed based on the Clohessy-Wiltshire equations. This planner provides a minimum fuel or time trajectory between two moving bodies in orbit.

All of the tasks in Phase III require moving the manipulator in the presence of obstacles. Because many Phase III problems involve moving objects, potential field methods[28], which are very fast, are employed.

In Phase IIIA, the work for grasping a moving object is divided into two basic levels. A low-level high-bandwidth controller attempts to track a virtual grasp frame on the objects – but it steers clear of joint limits, obstacles, and singularities. A higher level grasp planner continually selects (heuristically) the best virtual grasp frame on the object to track. In Phase IIIB, the controller will be expanded to include guarded moves (where contact with a fixed object is expected), force and impedance control, and position control with their hybrids.

Speech planning software starts with an unambiguous message created from the internal representation and attempts to construct a meaningful sentence in response. This is then sent to speech synthesis hardware. Several general-purpose single-sentence generators of natural language are moving toward full-scale commercial strength[29] and real-time generation[30], with the latter a candidate for EVAHR use.

Acting software provides low-level controllers of motors and other actuators. One important feature in EVAHR's acting software is visually directed sensing. Sensor parameters such as the field of view (FOV), the focus of attention (via pan/tilt devices), or the data acquisition rate can be dynamically selected in order to acquire richer information about the environment or objects of interest[31].

5. Phase IIIA Results to Date

Results from Phase II have been reported previously[32]. Some preliminary results from Phase IIIA have also been reported[25,33-38]. Results from Phase IIIA consist of evaluations of software architectures such as SOAR, along with computer simulation results of various portions of the software capabilities, including results allowing an estimate of the central processing unit (CPU) and communications requirements to achieve realtime grasp of floating objects. Results from KC-135 tests of unintegrated hardware and software subsystems are also given.

5.1 SOAR Evaluation for Phase IIIB

SOAR[16] was selected for study as a promising candidate system for the EVAHR planning system. SOAR is a symbolic AI architecture which emphasizes problem-solving, planning, and learning. It has been applied in numerous fields, such as education and training. As a production-based system, SOAR starts with an initial state of the problem and applies operators which make changes to the problem state to reach the goal state. Finding the sequence of operators to apply to the current problem state is the major challenge in its planning.

One major advantage of SOAR is its ability to learn by taking a new experience and saving the sequence of steps to the goal as a "chunk." This chunk is in the form of a set of production rules, and if the same scenario is encountered in the future, the associated chunk will execute without having to search for the correct sequence as it did initially.

From our experience with Hero-SOAR, a subset of SOAR for a Hero robot, we know that the reactivity of SOAR is an important capability needed to respond to the environment quickly. SOAR may be seen as a system with a planner, which plans in the traditional sense, yet with no actual data structure produced; a mechanism to execute the plan; and a fast replanning ability.

5.2 Phase IIIA Computer Simulation Results

Software modules for grasping of free-floating objects in a zero-g, 6-DOF environment have been described in previous sections. Results of performance testing of these modules as sub-systems are described in this section. The modules have also been integrated and tested in the orbital and KC-135 simulations[39], and these results are also described below.

5.2.1 Phase IIIA Computer Simulation Results - Uncluttered Search

The search is the first visual function to be performed when there is no knowledge about the location of an object of interest. It is carried out as follows[40,41]. EVAHR's front hemisphere is divided into concentric "rings," and each ring is further divided into sectors, each of which is enclosed by the FOV of the sensor. Each search starts from the center ring and spirals outward until an object is found. If an object is found, the search is terminated and the estimate of where the object is located is iteratively refined by adjusting the sensor gimbals toward the object and reducing the FOV until the object is centered and large in the image.

5.2.2 Phase IIIA Computer Simulation Results - Pose Estimation

Algorithms for image-based pose estimation have been implemented. Several objects were chosen for testing. These objects include some orbital replaceable units (ORU's), a star tracker, a jettison handle, and some wrenches.

To test the robustness of the software, 500 tests were run on each test object with actual poses of the object randomly oriented using a random number generator in (simulated) images. Noise was added to the "range" component of the image to test the sensitivity of the algorithms to noise. There were two indications from the test results: (1) Most estimation errors are less than 5 degrees (with up to 3-percent noise in range). (2) The performance of the pose estimation software gradually degraded with increasing noise in range measurements.

5.2.3 Phase IIIA Computer Simulation Results - State Estimation

The rotational state estimator uses intermittent delayed poses from the pose estimator software to provide the arm trajectory planner with current estimates of the target's rotational state at the rate of 100 Hz. The estimator utilizes an extended Kalman filter because of the inherent nonlinear nature of rotational dynamics. The effects of varying various parameters on the performance of the standalone rotational state estimator have been reported[34]. Testing on the integrated rotational state estimator shows it converges within four pose estimates (about 4 sec) and maintains error estimates of less than 3 degrees, which meets requirements.

The relative translational state estimator used for the KC-135 experiment does not use an inertial coordinate system. The equations describing the dynamics are nonlinear. Therefore, the estimator design is based on an extended Kalman filter. The results of its performance in the KC-135 simulator show an accuracy similar to that for the orbital case[42].

5.2.4 Phase IIIA Orbital Computer Simulation Results - Grasping Moving Objects

Integrated software testing in the orbital simulation has concentrated on and produced results in two areas: (1) determining the overall system performance against grasping different targets with random initial states and (2) determining the computational requirements for the pose estimation software, using rate and delay as parameters. In those tests, the following constraints hold: The target remains stationary in an optimal location for grasping; a grasp must be achieved in 15 sec. Grasp impact dynamics calculations are made to verify that the target is not knocked away during the grasp or by a prior collision with the arm. The EVAHR inertial state is assumed known. In the random initial state test suite, the target rotates in 3 DOF starting from a random initial orientation and velocity. Under these conditions, the system has achieved a >70-percent successful grasp rate for both objects tested. The state estimates have less than 1 inch and 5 degrees of error. An average time line of events in a typical successful grasp test is given in Table 2.

Table 2. Grasp Test Time Line.

Event	Time from start, sec
Translational state estimation initialized	0.21
Rotational state estimation initialized	4.67
Grasp command issued	4.78
Pose estimator feedback initiated	5.73
Grasp successful	10.91

The command to grasp is issued when the task sequencer sees that the rotational state has been initialized. The "pose estimator feedback" refers to predictions made by the state estimators which are used by the pose estimators to calculate the poses faster.

In the second suite of tests, the pose estimation rate and delay were varied. Figure 3 shows a snapshot from one of these tests. Results from this same set of tests show that pose estimation rate and delay also have a direct effect on the time-to-grasp in successful tests. Assuming pose estimation rate and delay of 0.1 sec, we were able to estimate that six i860 processors would be sufficient to achieve these rates and delays.

5.3 Aircraft Reduced Gravity Environment

Some microgravity research can be conducted inside an aircraft simulating space by flying vertical parabolic flight paths, but only for very limited amounts of time. During Phase IIIA we are flying a subset of the EVAHR Phase IIIB hardware and software aboard the NASA Reduced Gravity Program's KC-135 aircraft. This aircraft flies a series of parabolic trajectories resulting in approximately 15 sec of near microgravity (<.01-g) in the cabin during each parabola. The robotic arm, hand, vision sensor with pan/tilt system, and IMU of accelerometers and gyroscopes are attached to the floor of the aircraft. During microgravity, an object is released, tracked by the vision system, and grasped by the hand.

The objects to be used for grasping onboard the KC-135 aircraft range from simple to highly complex, but are limited to spheres or polyhedral surfaces. Some are lightweight mockups of actual objects used on orbit. Two of the objects are basically dumbbell-shaped objects with polyhedron-shaped masses at the ends. The more complex objects represent a battery from an Extravehicular Mobility Unit (EMU), a star tracker, and an ORU. All of these objects have a complex construction with multiple graspable points.

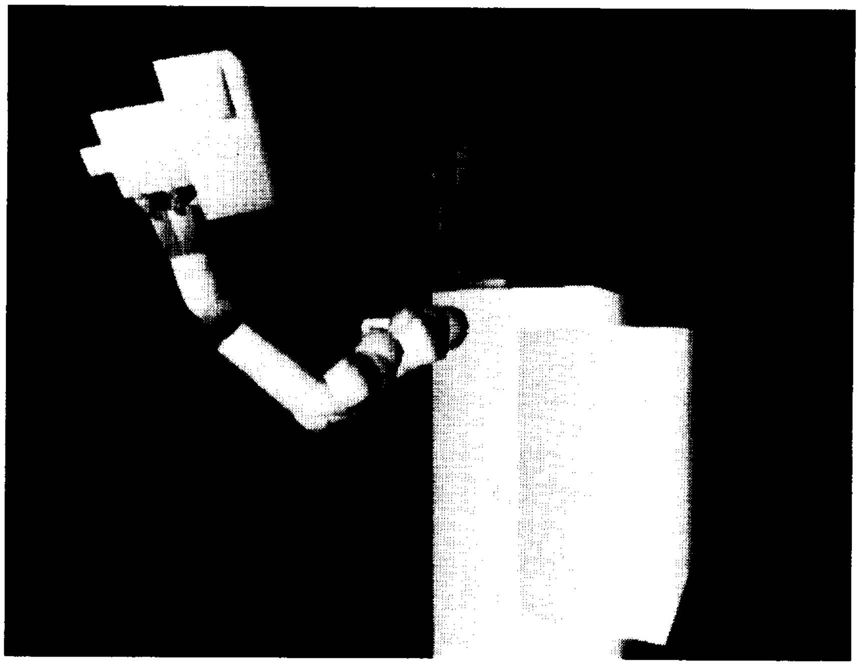


Figure 3. Orbital Simulation of EVAHR Grasping the "Backside" Handhold of Object.

On several KC-135 preliminary flights, data characterizing the reduced gravity was collected from an IMU placed on the cabin floor. Video recordings also were made of objects floating during the reduced gravity interval. The vertical acceleration fluctuated significantly about zero-g. Fluctuations between 75 mg and -75 mg were commonplace. These fluctuations caused the released object to accelerate toward either the ceiling or floor of the airplane. Lateral accelerations were also observed and were due to air turbulence, flight path corrections, or other effects.

An evaluation of 38 parabolas was performed, and the trajectory duration determined. This interval started when the target was released and continued until the target hit the inside of the airplane fuselage, was touched by personnel, or left the FOV of both video cameras. The results are presented in Table 3.

Table 3.– Duration of KC-135 Parabolas.

Duration of parabola, sec	Number of parabolas
7-8	2
6-7	5
5-6	6
4-5	2
3-4	2
≤3	21

These results, especially the trajectory durations, do not match well with the extrapolation to the KC-135 of time-to-grasp results from the orbital simulation presented above.

5.3.1 Phase IIIA Results – Hardware Evaluation From a KC-135 Flight

In a separate flight of the KC-135, we exercised the unintegrated hardware subsystems (except the stereo cameras) independently. All of the hardware is designed to operate in a 1-g environment and might behave differently in the KC-135 in microgravity or after the 1.8-g pullout at the bottom of the parabolas. Motions and operations representative of those that will be used in later object tracking and grasping evaluations were used in these tests. All equipment was determined to operate without measurable changes in behavior from that expected.

6. Conclusions

The need for crew help and retrieval of detached crew and equipment in space has been identified. Evaluation of the practical realization of a potential solution has passed several successful milestones but is still ongoing, with many of the critical developments yet to come. The potential solution described here is an initial attempt to build and understand a prototype of a supervised intelligent robot for use in space. It is also potentially useful in terms of the software architecture for many U.S. economy-related robot applications on Earth.

From our Phase II experience with both the interleaved sense-perceive-plan-act software architecture in a stationary environment and the reaction plans architecture in a dynamic, unpredictable simulated environment, we have concluded that (1) the success of the reaction plans approach argues for such a mechanism in an intelligent robot architecture to provide the capability for an appropriate quick reaction whenever perception understands the situation to provide an index into the correct reaction plan; (2) robot control architectures should be heterogeneous (different computational structures for planning and control); and (3) putting the AI planner at a high level of abstraction, which provides plans as goal-seeking guidance rather than direct control, and into an asynchronous mode are steps toward an intelligent robot architecture that can deliver safe behavior as well as goal-achieving behavior in a supervised intelligent robot. Our Phase IIIA experience to date in simulated real-time complex perception and grasping supports the reaction plan view. A way to appropriately integrate the two elements, AI planner and reaction plans, is needed which controls both. The three-tiered architecture may offer such an approach. Both the three-tiered architecture and SOAR are practical implementations of the mathematical theory of intelligent robots[43].

Both our Phase II and Phase IIIA results demonstrate that manipulation requires greater accuracy of sensing and perception than does mobility. Integrated testing with our Phase IIIA computer simulation has not only shown that we have a workable software design, but it has also afforded us systems engineering analyses supporting computer hardware design for achieving real-time complex perception processing (sensor to percept) and grasp control (percept to action) for freely moving objects.

Our future plans are first to complete the metrology of the manipulator and joint calibration of both vision-system-manipulator pairs. We are recoding the laser scanner pose estimation software to run in real time on the i860 network[44]. The tracker and translational state estimator are currently running in real time on i860's. The manipulator trajectory controller and grasp planner are running in real time on the transputer network. Grasp testing using targets mounted on the object-motion unit are being conducted in preparation for the KC-135 vision-guided grasping flights. Then, we have several moving object grasp evaluation flights to conduct. Phase IIIB developments are dependent on the selection of a final software architecture from the preliminary prototyping efforts which are underway using a set of crew helper tasks, scenarios, and computer simulation environments with human-injected, unpredictable events to assess the value of the many goal-planning and real-time reaction aspects of the supervised intelligent robot design.

7. Acknowledgments

This effort is internally supported by the Center Director's Discretionary Fund and by the Automation and Robotics Division of the NASA Lyndon B. Johnson Space Center.

8. References

1. J. D. Erickson et al., "An Intelligent Space Robot for Crew Help and Crew and Equipment Retrieval." International Journal of Applied Intelligence, accepted for publication in 1994.

2. J. D. Erickson et al., "SEI Planet Surface Systems Humans/Automation/ Robotics/ Telerobotics Integrated Summary." Document JSC-45100, NASA Johnson Space Center, Houston, TX, Mar. 1991.

3. J. D. Erickson, "Needs for Supervised Space Robots in Space Exploration." 1992 World Space Congress, IAF-92-0800, International Astronautical Federation, Paris, France, Aug. 1992.

4. J. D. Erickson et al., "Some Results of System Effectiveness Simulations for the First Lunar Outpost." 1993 AIAA Space Programs and Technology Conference, Huntsville, AL, Sept. 1993.

5. Bureau of Export Administration, "National Security Assessment of the U.S. Robotics Industry." Department of Commerce, Washington, D.C., Apr. 1991.

6. J. F. Engelberger, "Robotics in Service." Cambridge: The MIT Press, Jan. 1991.

7. Office of Technology Assessment, "Exploring the Moon and Mars: Choices for the Nation." Congress of the U.S., Washington, D.C., Aug. 1991.

8. R. P. Bonasso and M. G. Slack, "Ideas on a System Design for End-User Robots." Proceedings of SPIE 1829 Cooperative Intelligent Robotics in Space III, Nov. 1992, p. 352.

9. R. O. Ambrose, M. P. Aalund, and D. Tesar, "Designing Modular Robots for a Spectrum of Space Operations." Proceedings of SPIE 1829 Cooperative Intelligent Robotics in Space III, Nov. 1992, p. 371.

10. F. daCosta et al., "An Integrated Prototyping Environment for Programmable Automation." Proceedings of SPIE 1829 Cooperative Intelligent Robotics in Space III, Nov. 1992, p. 382.

11. W. L. Livingston, "TQM, Total Quality Management." Conference of the Quality Assurance Institute, Washington, DC, May 22, 1991.

12. E. Gat, "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots." AAAI-92, Proceedings of the 10th Natl. Conf. on AI, San Jose, CA, July 1992.

13. R. J. Firby, "Adaptive Execution in Dynamic Domains." Ph.D. Thesis, Yale University, 1989.

14. R. P. Bonasso, "Using Parallel Program Specifications for Reactive Control of Underwater Vehicles." Journal of Applied Intelligence, Kluwer Academic Publishers, Norwell, MA, June 1992.

15. J. E. Laird, A. Newell, and P. S. Rosenbloom, "SOAR: An Architecture for General Intelligence." Artificial Intelligence, Vol. 33, No. 1, pp. 1-64, 1987.

16. J. E. Laird et al. "RoboSOAR: An Integration of External Interaction, Planning, and Learning Using SOAR." Robotics and Autonomous Systems, Vol. 8, 1991.

17. S. B. Huffman and J. E. Laird, "Learning Procedures from Interactive Natural Language Instructions." Proceedings of the 10th Intl. Conf. on Machine Learning, June 1993.

18. E. S. Yager, "Resource-Dependent Behavior Through Adaptation." Ph.D. dissertation in computer science, University of Michigan, Ann Arbor, MI, Sept. 1992.

19. M. Schoppers, "Ensuring Correct Reactive Behavior in Robotic Systems." AIAA/NASA JSC Workshop on Automation and Robotics '93, NASA Johnson Space Center, Houston, TX, Mar. 24, 1993.

20. M. Schoppers, personal communication, July 1992.

21. C. Paul et al., "Reducing Problem Solving Variance to Improve Predictability." Communications of the ACM, Vol. 34, No. 8, 1991.

22. E. Durfee, "A Cooperative Approach to Planning for Real-Time Control." Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control, 1990, pp. 277-283.

23. P. Agre and D. Chapman, "Pengi: An Implementation of a Theory of Activity." Proceedings of the AAAI, 1987, pp. 268-272.

24. C. Elsaesser and T. R. MacMillan, "Representation and Algorithms for Multiagent Adversarial Planning." MTR-91W000207, MITRE Corp, McLean, VA, Dec. 1991.

25. C. H. Chien, "Multiview-Based Pose Estimation from Range Images." Proceedings of SPIE 1829, Cooperative Intelligent Robotics in Space III, Boston, MA, November 1992, pp. 421-32.

26. E. Gat, "On the Role of Stored Internal State in the Control of Autonomous Mobile Robots." AI Magazine, Spring 1993, pp. 64-73.

27. M. Magee, C. H. Chien, and T. W. Pendleton, "A Vision System Planner for the Extravehicular Activity Retriever." 1992.

28. O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." Intl. Journal of Robotics Research, Vol. 5, No. 1, Spring 1986.

29. E. H. Hovy, "A New Level of Language Generation Technology: Capabilities and Possibilities." IEEE Expert, p. 12, Apr. 1992.

30. T. Patten and D. S. Stoops, "Realtime Generation of Natural Language." IEEE Expert, p. 15, Oct. 1991.

31. D. Ballard, "Animate Vision." AI Journal, Vol. 48, No. 1, Elsevier, pp. 57-86, Feb. 1991.

32. J. D. Erickson et al., Technology Test Results from an Intelligent, Free-Flying Robot for Crew and Equipment Retrieval in Space." Proceedings of SPIE 1612, Cooperative Intelligent Robotics in Space II, Boston, MA, Nov. 1991.

33. M. L. Littlefield, "Adaptive Tracking of Objects for a Mobile Robot Using Range Images." Proceedings of SPIE 1829, Cooperative Intelligent Robotics in Space III, Boston, MA, Nov. 1992, pp. 433-443.

34. L. Hewgill, "Motion Estimation of a Freely Rotating Body in Earth Orbit." Proceedings of SPIE 1829, Cooperative Intelligent Robotics in Space III, Boston, MA, Nov. 1992, pp. 444-457.

35. K. A. Grimm et al., "Experiment in Vision-Based Autonomous Grasping Within a Reduced Gravity Environment." Proceedings of SPIE 1829, Cooperative Intelligent Robotics in Space III, Boston, MA, Nov. 1992, pp. 410-420.

36. C. H. Chien, "A Computer Vision System for Extravehicular Activity Helper/Retriever." Proceedings of SPIE Conference on Sensor Fusion and Aerospace Applications, Orlando, FL, Apr. 1993.

37. R. S. Norsworthy, "Performance Measurement of Autonomous Grasping Software in a Simulated Orbital Environment." Proceedings of SPIE 2057, Telemanipulator

Technology and Space Robotics, Boston, MA, Sept. 1993.

38. G. Anderson, "Grasping a Rigid Object on Zero-G." Proceedings of SPIE 2057, Telemanipulator Technology and Space Robotics, Boston, MA, Sept. 1993.

39. R. S. Norsworthy, "Simulation of the EVAHR/KC-135 for Software Integration/Testing." Proceedings of AIAA/NASA Conf. on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

40. M. L. Littlefield, "Real-Time Tracking of Objects in a KC-135 Microgravity Experiment." Proceedings of AIAA/NASA Conf. on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

41. E. L. Huber and K. Baker, "Object Tracking with Stereo Vision." Proceedings of AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

42. L. Hewgill, "Motion Estimation of Objects in KC-135 Microgravity." Proceedings of AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

43. G. N. Saradis, "Analytic Formulation of the Principle of increasing Precision with Decreasing Intelligence for Intelligent Machines." Automatica, Vol. 25, No. 3, pp. 461-467, 1989.

44. C. H. Chien, "An Architecture for Real-Time Vision Processing." Proceedings of AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space, Houston, TX, Mar. 1994.

LaRT

AIAA-94-1235-CP

556-63
207677
p. 7

Terrestrial Applications of NASA Space Telerobotics Technologies

N94-30582

Dave Lavery
NASA Headquarters

Introduction

In 1985 the National Aeronautics and Space Administration (NASA) instituted a research program in telerobotics to develop and provide the technology for applications of telerobotics to the United States space program. The activities of the program are intended to most effectively utilize limited astronaut time by facilitating tasks such as inspection, assembly, repair, and servicing, as well as providing extended capability for remotely conducting planetary surface operations. As the program matured, it also developed a strong heritage of working with government and industry to directly transfer the developed technology into industrial applications.

Program focus on user missions

Since its inception, the Telerobotics Program currently conducted by the Office of Advanced Concepts and Technology (OACT) has been closely coordinated with the NASA organizations which are the intended recipients of the developed telerobotics technology. This coordination takes place at multiple levels, with the potential user community technology needs expressed both formally and informally to OACT.

At the highest strategic level, OACT works with the user offices and industry to develop an annual Integrated Technology Plan (ITP) in support of the civil space program. The purpose of the ITP is to serve as a strategic plan for the OACT space research and technology programs, and as a strategic planning framework for other NASA and industry participants in advocating and conducting technology developments. The integration of strategic requirements, directions and goals for the Space Telerobotics Program is incorporated within the ITP process. The ITP is revised annually to reflect changes in mission planning, approval of new focussed and research base efforts, and progress in ongoing technology development efforts.

In addition to the formal submission of requirements from the user program offices to OACT via the ITP process, each user organization works informally with the Telerobotics Program at a more detailed level to transmit requirements to, and receive technology products from the program. This also includes gathering of requirements and opportunities from the terrestrial robotics industry, through an Industry Advisory Workshop held each year.

As these updated technology requirements are passed to the Telerobotics Program each year, the program is reassessed to determine the correlation between the requirements and the planned developments of the program. If appropriate, new tasks are initiated in the program to address new technology needs, or existing tasks may be re-targeted. At any given time, approximately 70% of the tasks within the program are targeted to address specific user requirements aligned with a specific planned mission (this is the "technology pull" portion of the program). The remaining 30% of the program is composed of tasks which address new innovative technologies. These technologies have been identified by the program as having a potential to significantly advance the state of the art, and worth investigating without a pre-identified user requirement (this is the "technology push" portion of the program).

The anticipated robotics requirements forwarded by the user offices to the Telerobotics Program during this past year are summarized in Figure 1.

In previous years, these tasks were organized within the program by technology sub-discipline, such as supervisory control, operator interface, planning and control, perception, etc. This organization was useful to the program participants and robotics community to ease understanding of the component technologies being developed. However, this organization made it difficult to identify how the tasks related to user needs. To resolve this situation the Telerobotics Program has been reorganized during the last year to better reflect the connections between the program tasks and the classes of planned user missions.

The Telerobotics Program has been restructured into three specific mission or application areas: on-orbit assembly and servicing, science payload tending, and planetary surface robotics. Within each of these areas, the program supports the development of robotic component technologies, development of complete robots, and implementation of complete robotic systems focussed on the specific mission needs. These three segments align with the application of space telerobotics to the class of missions identified by the

	Space Science:	Space Flight/Space Station:	Mission From Planet Earth*:
Requirements:	• low mass and volume planetary surface rovers • local rovers (<100m range) with multi-day lifetime • autonomous and semi-autonomous operation • improved system robustness • reduced operator command cycles • improved sensing and representation of state and worksite • miniaturized sensing and computing systems • simplified control approaches for small mobile systems • improved system dexterity and contact motion control • terrain mapping and matching	• telerobotic control system software • sensing and sensor fusion • simplified collision avoidance and trajectory planning • automated task planning and sequencing	• robotic vision and perception systems • advanced proximity sensing systems • advanced dexterous end-effectors • high-efficiency, long term lubrication for actuators
Applicable Missions:	• MESUR Pathfinder • MESUR Network • Mars Sample Return • Venus Landed Systems (Discovery) • Advanced robot surface systems	• Space Station Freedom maintenance • Space Station Freedom operations • on-orbit vehicle assembly and processing	• Artemis • First Lunar Outpost • Mars exploration • Permanently Manned Lunar and Mars Missions
Challenges:	• physical contact with planetary surfaces • uncertain knowledge of operating environment • long operational phases • radical reduction of life-cycle costs • multiple concurrent missions • very high data rate science payloads • high speed simulation of complex systems	• generalized solutions to 7-degree-of-freedom motion • multi-arm coordinated cooperative control • reduced on-orbit computational capability • computation or communications-induced time delays	• unknown dust contamination characteristics • low mass and volume constraints • long-duration pre-deployment storage • low- to no-maintenance operations

Figure 1: Current space robotics user requirements

potential space robotics user community (as summarized in Figure 1).

Two additional program segments have been defined to support the three focus areas. The Robotics Technology segment develops component technologies which have been determined to be of potential benefit in addressing multiple needs of the known robotics requirements. These elements of the program are typically long lead-time items, which may take many years to fully develop and bring to an appropriate level of readiness. This portion of the current program includes such elements as fundamentally new robotic joint designs, exoskeleton systems, fundamental robotic control theory development, and widely-applicable proximity sensor technology. The Terrestrial Robotics element of the program provides a mechanism for the application of developed technologies into terrestrial task environments. These tasks move the technologies developed in the other elements of the program from the laboratory setting into operational use, and take advantage of the relatively easy terrestrial access, well understood environments, and myriad problems to be solved to demonstrate the applicability of space telerobotics.

Links to other robotics programs

Throughout the life of the NASA Telerobotics Program, NASA has worked to build and maintain coordination with other government robotics programs, including those of the National Science Foundation (NSF) and the National Institute of Standards and Technology (NIST). These efforts include cooperative activities, collaborative research, and external transfer of NASA-developed robotics technology. These efforts have three purposes: to develop industrial applications of telerobotics technology, to apply telerobotics technology to terrestrial science and research efforts, and to strengthen intra-government coordination. Several of the activities are summarized below, beginning with the efforts targeting development of industrial applications of telerobotics technology:

- The Automated Manufacturing Research Program, conducted by NIST is investigating automation in factory-floor settings, and the relative advantages of improved work cells against more capable manipulation systems. NASA participates in the annual program review conducted by NIST, and coordinates with NIST to transfer NASA-developed robotics workcell technology into this effort.

- In previous years, NIST and the NASA telerobotics efforts have cooperatively developed several new technologies and architectures for the control of robotic systems. For example, the NASREM robot control architecture was jointly developed by NASA and NIST as a precursor to the NASA Flight Telerobotic Servicer program. The architecture is now used as a standard architecture definition methodology by many NASA, NIST and industry projects. NASA has directly supported NIST in several of these cooperative activities, with annual funding for robotics research reaching up to $1 million per year.

- The NASA and NSF robotics research programs have jointly co-sponsored the "Bilateral Exchanges on the Approaches to Robotics in the United States and Japan" conference, which conducted investigations into the methods, techniques and technologies used by government and industry to research and develop fundamental new robotics technologies. The outcome of this activity was publication of a manuscript which contrasted the approaches used in the United States and Japan, and which offered NASA and NSF insights into the content of the robotics development programs supported by MITI, NASDA, and several Japanese industries.

• The Advanced Research Projects Agency (ARPA) has selected the Langley Research Center robotics program as one of their technical agents in the area of robotics. Under this agreement, LaRC and DARPA cooperatively issue university research grants to sponsor the development of innovative new robotics technologies, as well as increase robotics educational expertise in the United States.

• The program has maintained close ties with the U. S. space robotics industrial community, and monitored industrial developments of potential applicability to the NASA space robotics and planetary rover research efforts. For example, the Martin-Marietta Corporation participates in the Telerobotics Intercenter Working Group, and in technical program reviews and assessments such as the Space Systems Technology Advisory Council. This coordination facilitates the transfer of NASA-developed technologies to the space robotics industry, and aids in the rapid application of these technologies to terrestrial manufacturing and automation problems.

• The program coordinates with several robotics industry advisory and technology interchange groups, to facilitate the transfer of NASA-developed technology to the industrial community and receive comments on the overall direction and focus of the program. One such group is the Space Automation and Robotics Technical Committee (SARTC) of the American Institute of Aeronautics and Astronautics which meets three or four times annually with the charter of disseminating information about space automation and robotics and promoting the technology to industry, academia, and government. The SARTC is composed of industry representatives from the aerospace community, as well as government and academia.

Some of the efforts which target application of telerobotics technology to terrestrial science and industry efforts are listed below:

• Several programs sponsored by NSF both sponsor and utilize telerobotics and robotics technology research and development. In 1992 NASA and NSF cooperated in conducting the Mt. Erebus Explorer project, a project to deploy a robot into the interior of a volcano crater in the Antarctic. This project, conducted as part of the Telerobotics Program and the Antarctic Space Analog Program, demonstrated innovative new robotics technologies developed by NASA. It is anticipated that this project will spawn several new activities which may revolutionize volcanic sample collection and lead to significant new applications of robotics in terrestrial field science operations. This project is being continued with the United States Geological Survey, and will deploy the Dante robot to a volcano in Alaska in the summer of 1994.

• In addition to the involvement with the NSF Polar Programs Division (which cooperated with the Mt. Erebus Explorer project), NASA is currently negotiating with the NSF Oceans Division to investigate the potential for application of NASA-developed robotics technology to underwater science sampling operations. Of particular interest is the underwater Remotely Operated Vehicle (ROV) technology which NASA developed and demonstrated under the Antarctic sea ice with the cooperation of NSF in 1992. Additional negotiations are underway with the NSF Information, Robotics and Intelligent Systems Division to jointly sponsor robotics research and investigate opportunities for transfer of NASA-developed robotics technologies to NSF grantees and research programs.

• The robotics laboratories at the Jet Propulsion Laboratory have been working with Computer Motion, Inc. to develop technologies for applications where human ability to perform a task is limited by human dexterity and physical capabilities. One specific application has been in minimally invasive laproscopic surgery. This type of medical procedure makes use of remote cameras, known as laproscopes, which are typically held by an assistant to the surgeon during a procedure. The assistant has control of the surgeons field of view, and the surgeons performance is often limited by the efficiency of communication with the assistant. To address this problem, the project has developed the Automated Endoscopic System for Optimal Positioning (AESOP), a robotic assistant which holds the laproscope and is guided by the surgeon with a foot- and/or hand-controlled interface. Thus the surgeon is able to gain control of the viewfield by direct coordination between himself and a robotic assistant.[1]

• JPL has also worked with Cybernet Systems Corporation to develop the PER-Force hand controller which manipulates robots or objects by "feel". this small backdrivable robot is combined with advanced machine vision processing and enhanced computer generated visual/tactile force feedback cues to enable an enhanced interface for the use on hazardous environment operations. This system has been implemented with a goal of integrating it within the manufacturing environment and tasks which have no immediate solution with hard automation or changes in methodology or workcell design. One example application being developed is pick-and-place operations for automobile transmission packing.[2]

• As an offshoot of work sponsored by the program, the Stanford University Aerospace Robotics Laboratory and Real-Time Innovations, Inc. have developed ControlShell, a next generation CASE framework for real-time system software development. ControlShell includes many system-building tools, including a graphical flow editor, a component data requirement editor, a state-machine editor, a distributed data flow manager, an execution configuration manager, an object database and a dynamic binding facility. ControlShell is being used in several applications, including the control of free-flying robots, underwater autonomous vehicles, and cooperating-arm robotic systems.[3]

• NASA has teamed up with Limbs of Love and a group of medical and prosthetics specialists, prosthetics users, insurance industry representatives, and university researchers to identify research objectives in prosthetic limbs. As part of this effort, the NASA Johnson Space Center has been actively working with Rice University to improve dexterous hand design and to develop a method for myoelectric control of multifinger hands. In theory, myoelectric control of robotic hands will require little or no mechanical parts and will greatly reduce the bulk and weight usually found in dexterous robotic hand control devices. An improvement in myoelectric control of multifinger hands will also benefit prosthetics users.[4]

This list is not exhaustive, but is a representative cross-section of the type of activities onducted by the NASA Telerobotics Program and other government organizations. Additional efforts have extended this coordination to industrial telerobotics research programs, to aid in the transfer of government-developed technology to the U.S. commercial/industrial robotics community. These efforts use two mechanisms to transfer the technology developed by the program.

The first mechanism pairs NASA researchers and commercial developers together to develop space telerobotics technology which is based on commercially-available products. As the terrestrial systems are extended to address the needs of the space telerobotics program by the researchers, the commercial partners are able to identify markets and applications for dual-use implementations of the new technologies, and rapidly incorporate them into new product lines. An example of this is the development of the "phantom robotic control" technology developed by JPL under the Advanced Teleoperation project. This technology has been developed as an extension to the commercial Interactive Graphics Robot Instructional Program (IGRIP) software package from Deneb Robotics. JPL has worked with Deneb to smoothly integrate the extension into the IGRIP package, and negotiated a mechanism to provide this extension to Deneb for commercialization. Deneb has identified a new need for this technology, beyond the original application of space telerobotics, and plans to incorporate the extension into their commercial product line.

The second mechanism pairs NASA researchers with commercial developers to work jointly on the application of space telerobotics technologies to terrestrial problems. The commercial partners bring existing terrestrial robotics systems and capabilities into the project, and work jointly with NASA researchers to improve these systems through the application of space telerobotics technology. An example of this is the Hazardous Materials Handling Robot (HAZBOT) project at JPL, which is being conducted with the partnership of Remotec, Inc, and which is addressing the problem of hazardous chemical spill incident identification and mitigation through the use of robotics. JPL and Remotec worked to apply technologies developed by the Telerobotics Program to improve the off-the-shelf Remotec "Andros" mobile robot to satisfy the unique needs of the HAZBOT project. Several of the specific techniques and mechanisms developed during this process have been delivered back to Remotec for incorporation within their commercial product line.

The program has similar interactions with other members of the U.S. industrial robotics community, such as Robotics Research and Oceaneering. The current program plans include expanding these efforts to include a larger percentage of the U.S. robotics industry.

Summary

NASA has put in place a comprehensive planning process which fully integrates the development of new technologies with stated user requirements and defined application areas. The ongoing space automation and robotics program is focused on responding to the needs and requirements of internal agency users, but also produces significant spin-off products which are passed on to other government and industrial users for terrestrial utilization. The program fully involves agency users, NASA field centers, industry and academia in both the development and end-use of the developed A&R technologies.

---

[1] Neville Marzwell, Darrin Yecker and Yulun Wang, Force-Controllable Macro-Micro Manipulator and its Application to Medical Robotics, Technology 2003, Anaheim CA, December 1993.

[2] Neville Marzwell, Charles Jacobus, Thomas Peurach and Brian Mitchell, Use of Interactive Computer Vision and Robot Hand Controllers for Enhancing Manugacturing Safety, Technology 2003, Anaheim CA, December 1993.

[3] Stanley Schneider, Vincent Chen and Gerardo Pardo-Castelote, ControlShell: A Real-Time Software Framework, Proceedings of the International Conference on Robotics and Automation, IEEE, May 1994.

[4] Clifford Hess, Larry Li, Kristin Farry and Ian Walker, Application of Dexterous Space Robotics Technology to Myoelectric Prostheses, Technology 2003, Anaheim CA, December 1993.

AIAA-94-1236-CP
N94- 30583

60 78

ON-ORBIT SPACECRAFT SERVICING
AN ELEMENT IN THE EVOLUTION OF SPACE ROBOTICS APPLICATIONS

S57-63

P. 10

Carl J. Anders
Senior Engineer, Advanced Systems
Spar Aerospace Ltd.
Advanced Technology Systems Group
9445 Airport Road
Brampton, Ontario
Canada L6S 4J3
Office: (905) 790-2800 FAX: (905) 790-4400

Claude H. Roy
Senior Marketing Specialist, Space Servicing
Spar Aerospace Ltd.
Advanced Technology Systems Group
9445 Airport Road
Brampton, Ontario
Canada L6S 4J3
Office: (905) 790-2800 FAX: (905) 790-4452

ABSTRACT

This paper addresses the renewed interest in on-orbit spacecraft servicing (OSS), and how it fits into the evolution of space applications for intelligent robots.

Investment in the development of space robotics and associated technologies is growing as nations recognize that it is a critical component of the exploration and commercial development of space. At the same time, changes in world conditions have generated a renewal of the interest in OSS. This is reflected in the level of activity in the U.S., Japan and Europe in the form of studies and technology demonstration programs. OSS is becoming widely accepted as an opportunity in the evolution of space robotics applications. Importantly, it is a feasible proposition with current technologies and the direction of ongoing research and development activities. Interest in OSS dates back more than two decades, and several programs have been initiated, but no operational system has come on line, arguably with the Shuttle as the exception.

With new opportunities arising, however, a fresh look at the feasibility of OSS is warranted. This involves the resolution of complex market, technical and political issues, through market studies, economic analyses, mission requirement definitions, trade studies, concept designs and technology

demonstrations. System architectures for OSS are strongly dependent on target spacecraft design and launch delivery systems. Performance and cost factors are currently forcing significant changes in these areas. This presents both challenges and opportunities in the provision of OSS services.

In conclusion, there is no question OSS will become a reality, but only when the technical feasibility is combined with either economic viability or political will. In the evolution of space robotics satellite servicing can become the next step towards its eventual role in support of planetary exploration and human beings' journey out into the universe.

1. SPACE ROBOTICS

Past and Present

The first space-based robotic arm was the Shuttle Remote Manipulator System (SRMS), better known as the Canadarm. Space mechanisms date back almost to the first excursions into orbit in the late 1950s, early 1960s, and the first primitive `robotic elements' were surface samplers on the planetary probes. The U.S. moon rover and unmanned Soviet rover should also receive honourable mention. However, the first true multi-degree-of-freedom robotic manipulator was the SRMS, launched on the Space Transportation System (Shuttle) in 1981. To this day it remains the only operational space robotic arm. (*See Figure 1.*)

A number of technology demonstrations in space robotics are proposed or have been performed, but the next major development in the field will be the deployment on the Space Station of the Mobile Servicing System (MSS), currently scheduled to begin operations in 1998. Elements of this system include the Space Station Remote Manipulator System (SSRMS) - a derivative of the SRMS, and the Special Purpose Dexterous Manipulator (SPDM) - a pair of arms equipped with special end effector tools, and designed for greater dexterity than the SSRMS. The MSS also includes a Ground Segment. (*See Figure 2.*)

Looking to the Future

There is a general consensus that robotics will play a major role in the exploration of space frontiers in the next century. Roles envisaged include: the construction of orbiting platforms, for manned occupation, materials storage, scientific experimentation and manufacturing; the assembly of planetary and interstellar expedition spacecraft; the construction and maintenance of habitats, and the construction, maintenance and operation of mining and industrial facilities on the moon and planets. Although manned presence will be desirable and necessary in many instances, safety and cost considerations are factors in moving space robotics development away from local manned operation towards teleoperated systems. Remote operation removes the risk to human life and eliminates the need for costly life-support systems. Even if astronauts are at the work site, the use of teleoperation can free up their valuable time for other tasks. Even more significant is the shift in thinking towards fully or semi autonomous systems, prompted by technology developments in such areas as sensing, artificial intelligence and predictive systems. This enables smart space robotic systems to function with minimal support from space or ground-based human operators. The penalty with the use of autonomous over teleoperated systems is the added weight and cost of placing high powered computational capabilities onboard the servicing vehicle.

Technology Issues

The application of teleoperation and autonomous robotic systems is by no means limited to space. Terrestrial tasks in isolated locations or hazardous environments have very similar requirements. This overlap extends to the critical technologies. For example, the two major problems inherent in the teleoperation of space systems are bandwidth limitations and time delays in the transmission and relay of signals (latency). These are the same issues found in deep sea (untethered) robotics applications.

The bandwidth limitations impose constraints on the quantity of data that can be transmitted between the ground station and the space system. The operator and the supporting computer facilities may have to work with incomplete or low resolution information.

It is generally accepted that latency greater than 0.25 to 0.5 seconds make real-time control by a human operator difficult if not impossible, and may demand a `move-and-wait' approach. This can be an acceptable control strategy in non-critical tasks, but has a high level of risk in complex, fast moving or close proximity operations.

An alternative to resolving the bandwidth and latency restrictions associated with teleoperation is to employ onboard autonomous control of space systems. The challenge then becomes design and installation of controllers with sophisticated sensing and highly developed intelligence. Realistically in the near and medium term such operations will still require supervision and optional override by a human operator to handle unforseen contingencies.

The trend towards remote teleoperation and local autonomy in the control of robotic systems matches the projected long term needs of space exploration, and in the near term it opens the door for satellite servicing.

2. ON-ORBIT SATELLITE SERVICING

On-Orbit Spacecraft Servicing (OSS) is simply the provision of space-based services to orbiting craft. The interest in spacecraft servicing with its substantial benefit potential has led to the performance of a number of studies, proposals and programs. Despite this activity, and the success of Shuttle based on-orbit robotic operations, an OSS system has yet to be implemented.

A Brief History

The possibility of servicing spacecraft in orbit has generated considerable interest in the space community for over 20 years. A primary objective in the development of the U.S. Space Shuttle was to reduce space program costs by replacing expendable launch vehicles (ELVs) with a fully reusable system capable of maintaining, refurbishing and upgrading payloads. The original Shuttle concept included a space tug for the purpose of transporting satellites to and from a Shuttle-achievable orbit. However, projected high development costs forced a descope of the Shuttle capabilities, and the resulting configuration (the one flying today) has significantly reduced servicing capabilities, and no means of accessing high altitude orbits or high inclination low earth orbits (LEOs).

Many LEO spacecraft are accessible to the Shuttle, and several missions included the rescue and repair of scientific and communications spacecraft. The Shuttle program has been highly successful from a technical viewpoint in demonstrating OSS possibilities despite the constraints on the scope of its market and the failure of generated revenue to cover the mission costs.

The Challenger accident in 1986 had a dramatic impact on Shuttle operations with repercussions felt throughout the space industry. A direct result on the Shuttle was a greatly reduced launch capacity and very stringent mission restrictions forcing a shift in spacecraft designs from Shuttle launched systems to ones compatible with ELVs. With concurrent advances in satellite miniaturization, smaller, less expensive, expendable platforms became more attractive. Through the late 1980s only a few large, complex platforms such as the Hubble Space Telescope were specifically designed to be serviceable by the Shuttle.

Access to a wider range of orbits including geosynchronous (GEO) was still desirable and a number of studies were commissioned to address that. None led to operational systems. Often, with a preconceived system architecture in mind, they focused on a single specific market. These markets were not always well chosen, and the systems lacked the flexibility to adapt to market changes. The most recent examples were the Orbital Maneuvring Vehicle (OMV) program and the Satellite Servicing System (SSS).

A New Perspective

The history of space servicing has been somewhat inauspicious, but there is more interest than ever before with initiatives underway in the U.S., Canada, Japan and Europe. These involve market studies, economic analyses, mission requirement definitions, trade studies, concept designs and technology demonstrations in preparation for the development and implementation of pay-for-service systems. This revival of interest reflects a changing market. The old views and perspectives on servicing markets may no longer be applicable, and new opportunities are arising. An example of this is the shift towards Smallsats, and the first major application - Low Earth Orbit (LEO) constellations of communications spacecraft.

The unprecedented number of recent launch failures has kept the satellite insurance rates in a constant state of flux. The launch vehicle industry, itself a critical parameter in the OSS equation, is in transition. Increased commercialization promises reduced launch costs and greater availability. These and other factors will impact both the servicing and serviced systems, and suggests that a fresh look at OSS is warranted.

Services

The traditional concept of OSS is spacecraft repair or refurbishment, but services such as refuelling to extend operating lives, and the transportation of `dead' vehicles into graveyard orbits are becoming increasingly important.

The term service can refer to any operation performed by one vehicle (servicer) on another vehicle (target or object). The primary possibilities for OSS include inspection, mechanical intervention, and repair/refurbishment activities. Payload upgrades may be offered through Orbital Replacement Unit (ORU) changeout. Refueling can be an important service since fuel capacity is the primary life-limiting factor for on-orbit satellites. Orbit transfer is a possibility for injection of a spacecraft into its correct orbit after apogee motor failure, correction of a drifting spacecraft, or placement of a `dead' spacecraft into a graveyard orbit. Forced reentry into the earth's atmosphere may be a disposal option in LEO. Spacecraft harvesting, placement and retrieval of experiments, recovery of data, and spacecraft reconfiguration are also potential OSS services. On-orbit construction of space structures can replace the need for humans to work on such physically demanding tasks in a hostile environment. A distinct (if less likely) application is that of space debris clean-up.

System Architecture Trades

The development and implementation of an optimum commercial system involves the resolution of complex market, technical and political issues. Interwoven with each of these are the many economic factors that ultimately determine whether spacecraft owners and operators are willing to pay for on-orbit services. From another perspective, satellite design life is balanced against payload obsolescence. The costs of making spacecraft serviceable must be weighed against service vehicle capability, and external influences such as fluctuating insurance rates.

The subject of technically and economically viable architectures for OSS is complicated by the fact that there are several potential markets, each of which presents a set of mission architecture trades. A satellite market can be segmented according to satellite type or function, and orbit. A further distinction is customer type. For example, commercial, civil government and defense communication satellites can have very different service requirements. An OSS system capable of satisfying a particular market has a distinct set of requirements, but many possible configurations. Not only will market shifts have a significant effect on the system architecture, but also on the cost and availability of the technology and hardware for the OSS system itself.

OSS system architectures can be assembled by considering the service base, service vehicle, control type and resupply vehicle. The system can be ground-based (service vehicle launched on demand), or space-based (service vehicle resident in space). The service vehicle may be expendable (short mission), or reusable (multi-mission). The control can be ground-based, space-based, or local autonomous. If replacement of consumables is required, the resupply vehicle may also be either expendable or reusable.

One of the simplest system architectures that can be envisaged is a small autonomous spacecraft capable of visiting satellites and performing a non-intrusive inspection. An extension of this could be to add some robotic capability for simple mechanical tasks. A much more complex concept is that of a multi-purpose vehicle with the ability to inspect, repair, refuel or transport a satellite, perhaps operating from a space-based storage depot that is resupplied from the ground. Another consideration is that an OSS system does not have to be based in space. It is conceivable to propose a quick response ground-based service system that would be launched on demand at short notice, an approach that may be preferred as launch costs decrease. The possibilities are almost unlimited.

OSS Viability

The conditions necessary for OSS to become a reality are the maturity of the required technology in conjunction with either economic viability or political will. The level of technology development particularly in the key area of space robotics is suitable for the implementation of an OSS system. The other half of the equation is not so clear.

The revenue-generating potential of OSS has been demonstrated by NASA's Shuttle-based repair and recovery of satellites. A good example of this was the reboost of Intelsat VI into its correct orbit in May 1992. Intelsat saved $200 million over the alternative option to manufacture and launch a replacement spacecraft and lose revenue during the accompanying delay, despite their recovery expenses of $147 million. This revenue level is not sufficient to finance a Shuttle mission generally costing in excess of $500 million, and NASA failed to recover its own costs. It seems reasonable though to assume that with an appropriate commercial OSS system in place under such circumstances, the Intelsat VI recovery could have been effected with substantial benefits accruing to all parties.

A NASA Group Task Force re-evaluation of its Shuttle program priorities and objectives after the Challenger loss concluded that in spite of the considerable need for spacecraft servicing, it does not serve NASA's interests to actively pursue the market using the Shuttle, for the primary reasons of safety and cost. This opens the door for the introduction of a dedicated OSS system designed along commercial lines.

There is no question that OSS has the potential to be a commercially viable and profitable business. The economics however are complex and represent a major hurdle in the transition from OSS concept studies to development of an operational system. The US Department of Commerce forecasts a commercial space market in the billions of dollars. The challenge facing a potential OSS developer is to select a market and define a system architecture that will offer sufficient potential returns at an acceptable risk level. Another obstacle to be overcome is the implementation of a commercially viable OSS system when existing spacecraft are not designed to accommodate servicing. The possible solutions to this problem are: (a) to develop an initial system providing limited services to existing spacecraft; (b) to develop an OSS system in conjunction with a new generation of serviceable spacecraft; (c) to respond to any future political legislation, for example the introduction of a policy on satellite recovery or disposal which may result from the space debris problems.

Concern with regard to orbital crowding and space debris is mounting. The situation in both LEO and GEO locations is becoming critical, and the move towards Smallsats will only accelerate the problem. Resolution of this, though difficult to quantify on a purely economic level, could be a catalyst in bringing OSS into being.

The design, manufacture, launch and operation of space systems will always be a costly undertaking. The benefits associated with repair and refurbishment will therefore continue to be attractive. The implementation requires an economically or politically viable concept for servicing satellites. The balance may eventually be tilted in favour of OSS if spacecraft interfaces were standardized, or if international legislation were enacted to enforce or encourage the recovery or disposal of spacecraft. Assessment of the trends and the forces at work, it seems safe to state that space servicing will become a reality - it is just a matter of time.

3. SPACE ROBOTICS AND OSS

Overview

OSS will require the next generation of autonomous, semi-autonomous or teleoperated robotics with advanced ground control - a natural progression from the manned robotics technology of the Space Shuttle (SRMS) and Space Station (MSS) programs. Indeed, looking to the future, since automated robots will be a key element in planetary exploration programs targeted for the next century, OSS development would appear to be to be a strategically astute policy for the space community. Participation in concept studies, technology demonstration programs, and development programs for satellite servicing can potentially facilitate the ongoing development of space robotics and its associated technologies. A strong case can be made that international collaboration is necessary in the evolution of OSS because of the anticipated high system implementation costs. It also promotes cooperation in the establishment of roles in a multinational effort that will produce global benefits.

Technology Discussion

The technologies associated with space robotics that are key to the development of an OSS system are discussed below.

Robotics, Tools & Mechanical Interfaces

In the broadest sense, unmanned spacecraft are themselves robots, but in this context the definition applies to manipulators, tools and devices for OSS tasks such as spacecraft capture, handling, berthing, end effector positioning, mechanical intervention, repairing, refurbishing, ORU changeout etc.

Vision Systems & Sensing

Operational requirements for OSS include target identification, ranging, 3-D mapping, multispectral sensing, lighting, photogrammetry. Much applicable work is being done in the area of hazardous waste remediation.

Telefunction

Telefunction refers to all aspects of the control of an advanced space system from the ground. It includes teleoperation, ground/space partitioning, predictive displays, data processing and much more.

Autonomous Rendezvous and Docking

This is really the combined application of many other technologies such as automated and remote controlled robotics, vision systems, mechanical interfaces, telefunction, power & data transfer, communications.

Communications

Satellite/ground communications is a critical area for OSS due to the need for transmission of data for control of complex tasks. As stated, data bandwidth limitations and signal latency are key issues.

4. CONCLUSIONS

The level of space robotics activity within the global community supports the view that it is a critical technology, and will continue to be so as we move towards the exploration of our solar system and beyond. The U.S., Canada, Japan and Europe have at the same time independently and almost simultaneously identified on-orbit spacecraft servicing (OSS) as a promising endeavour, and one that provides an opportunity for application of this robotics technology. The major space industries are pursuing OSS concept studies, technology demonstrations and program definitions. It is widely accepted that these initiatives will lead to the establishment of multinational alliances in future OSS programs. Furthermore, it is proposed that OSS will be the first major commercial application of remote controlled and autonomous space robotics.

REFERENCES:

1. Satellite Services Systems Analysis Study - NASA Johnson Space Center Contract NAS 9-16121, Lockheed Missiles and Space Company, August 1982

2. Satellite Services Systems Analysis Study - NASA Johnson Space Center Contract NAS 9-16120, Grumman Aerospace Company, March 1983

3. Satellite Servicing Mission Preliminary Cost Estimation Model - Science Applications International Corporation, NASA Johnson Space Center Contract NAS 9-17207, January 1987

4. Satellite Servicing Systems - Book of Issues, Engineering Directorate, Flight Projects Engineering Office, NASA Johnson Space Center, August 1987

5. Long-Term Orbital Lifetime Predictions - National Technical Information Service, 1990

6. Telerobotics, Automation, and Human Supervisory Control, Thomas B. Sheridan, MIT Press, (C) 1992 Massachussets Institute of Technology

7. NASA Report of the Group Task Force on Satellite Rescue and Repair, September 1992

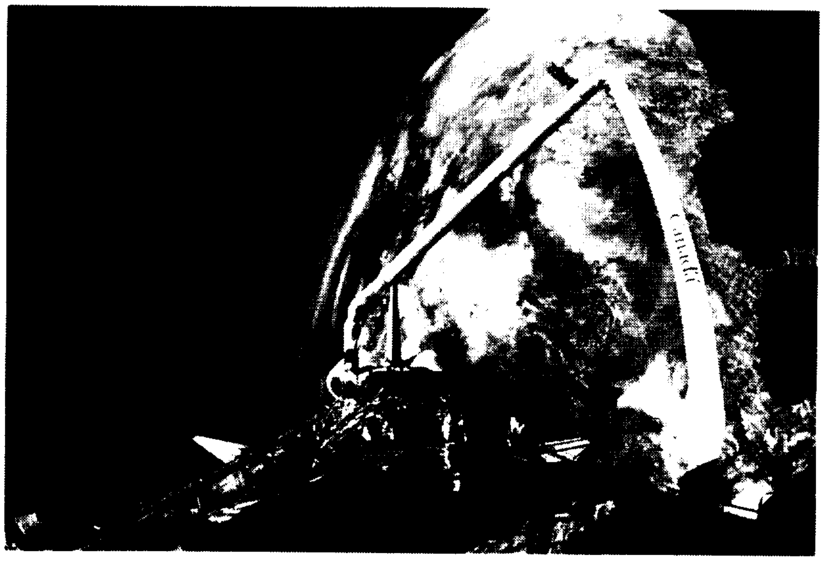


Figure 1. SRMS On-Orbit

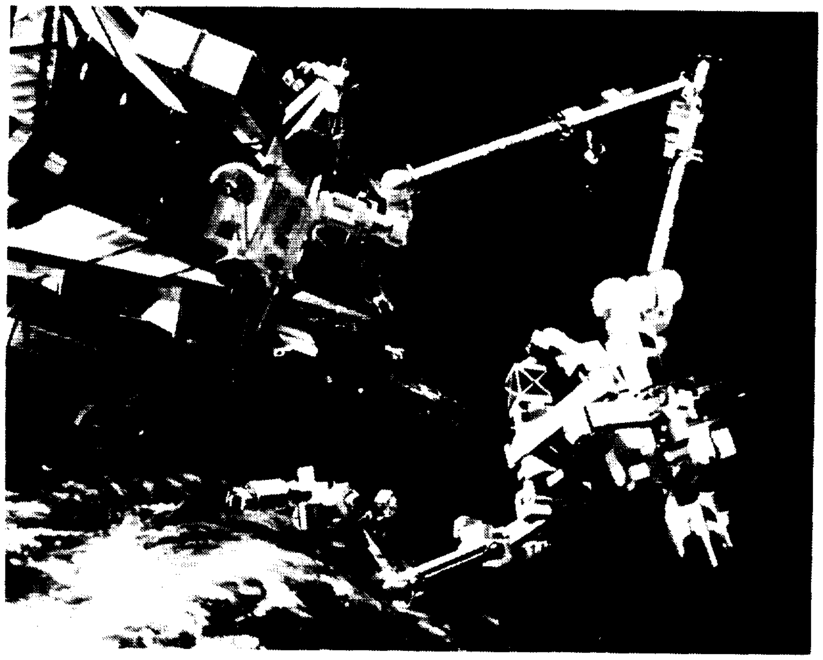


Figure 2. MSS Concept Illustration

[illegible]

AIAA-94-1237-CP

N94-30584

A Modular Artificial Intelligence Inference Engine System (MAIS) For Support Of On Orbit Experiments

Thomas M. Hancock III

New Technology Incorporated
700 Boulevard South, Suite 401
Huntsville, Alabama 35802
hancock@kiwi.msfc.nasa.gov

[illegible]

Abstract

This paper describes a Modular Artificial Intelligence Inference Engine System (MAIS) support tool that would provide health and status monitoring, cognitive replanning, analysis and support of on orbit Space Station, Spacelab experiments and systems.

Introduction

Most experiments flown on Spacelab to date have required considerable support from Mission and Payload Specialist on orbit and console operators and experiment investigation teams on the ground. During the Space Station era, the volume of experiments on orbit necessitates the development of an autonomous system for experiment management, monitor, operation and support.

This paper describes a reusable Modular Artificial Intelligence Engine System (MAIS) that will be flexible, have a low unit cost and will be reconfigurable to meet the needs of different experiments or systems. The MAIS would provide Mission and Payload Specialist with health and status monitoring, recommended actions to be taken and/or analysis of situations based on conditions and the problem encountered. The basic design of the system will allow "new artificial intelligence (AI) capabilities" to be added as needed. Reconfiguration for a new experiment will be accomplished by changing the heuristics and rules that will form the basis of scripts and can be accomplished as often as required.

This paper will also define the goals and objectives of the MAIS and the operational environment.

Operational Concept

Before flight or utilization on orbit, rules and scripts are developed or modified that support a particular experiment. An assessment of AI capabilities is performed, and if required, the software architecture of the MAIS is reconfigured. Interfaces to the experiment are established and graphical user interface displays are created to the user requirements. Limits, signal exceptions and health and status monitor rates are set. The MAIS tool and the experiment are started.

If sensor inputs, experiment signals or health and status updates indicate a problem, exceeded limit or a deviation of a pre-set plan, the tool will access rules and if necessary, apply heuristic or inference techniques (or other techniques as required) to identify and correct the problem. The tool will report the error and identify its most likely source, the applicable rules and what scripts were utilized to correct or replan around the problem. If necessary, the tool will perform as much of an analysis as possible following the steps previously outlined and when completed signal for human intervention.

Goals and Objectives

The high level goals and objectives of the MAIS are:

- Reduce experiment/ system support required by members of the Space Station/ Spacelab crew and ground operations personnel
- Reduce risk by managing and monitoring experiments continuously
- Reduce risk by providing an analysis of problems and recommending/ taking corrective actions quickly
- Provide a reusable tool that can be reconfigured for different experiments/ missions
- Provide analysis of the situation based on conditions and problems encountered
- Provide a technology transfer "autonomous system/process management, monitor and control" for researchers, product development or production environments

General Description

The MAIS will be composed of the following hardware and software elements:

1. Hardware module (containing CPU, memory, power, experiment interface and user interfaces)
2. Modular Inference Engine (design of the code will permit the introduction of additional AI capabilities (deduction, forward and backward chaining etc., as required))
3. Rule sets
4. Scripts
5. Graphical User Interface

Architecture

MAIS architecture is designed around an inference engine (expert system) which employs heuristics and rules to monitor and respond to sensor inputs or signals from an experiment through an experiment interface. MAIS user interface will display status data from the experiment and actions taken by the system in response to values exceeding limits or being non-responsive. The user interface will also support human intervention in the tool, its actions or the experiment at any time (see Figure 1).

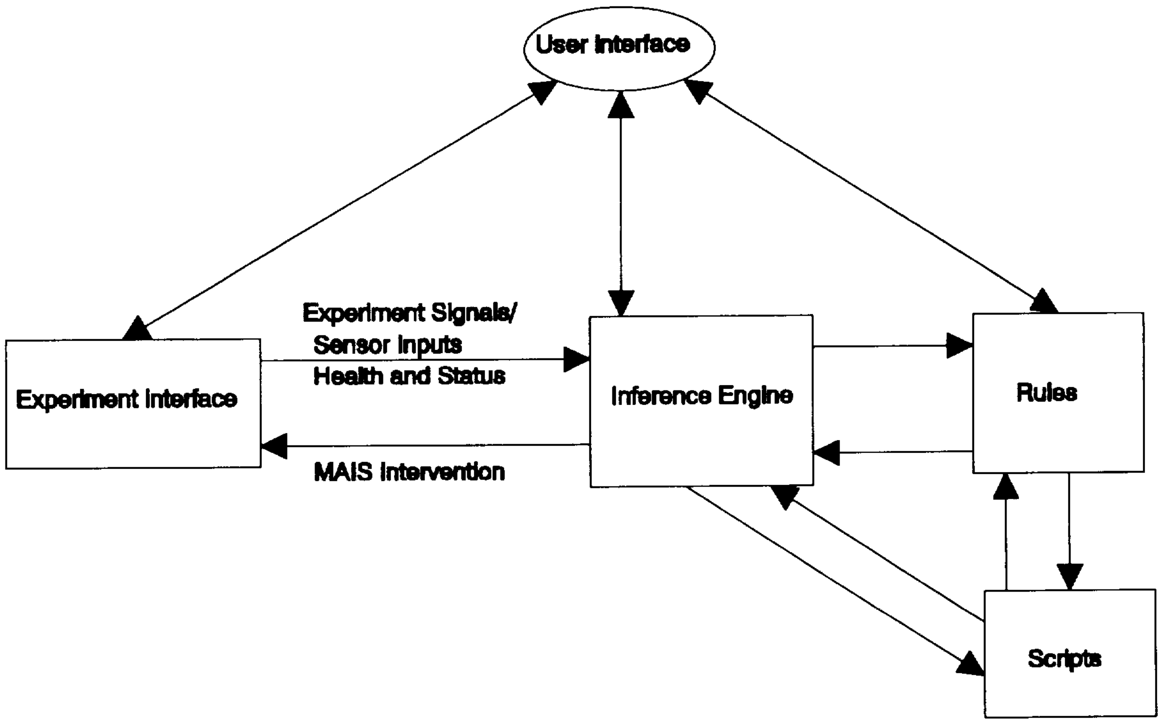


Figure 1. MAIS Architecture

Modular AI Design

One of the advantages of this tool will be the ability to add new AI capabilities to support the needs of different experiments. Some experiments will require deduction techniques, data search techniques or more extensive heuristics. A well designed modular system will support the "plugging in" of new capabilities (see Figure 2).

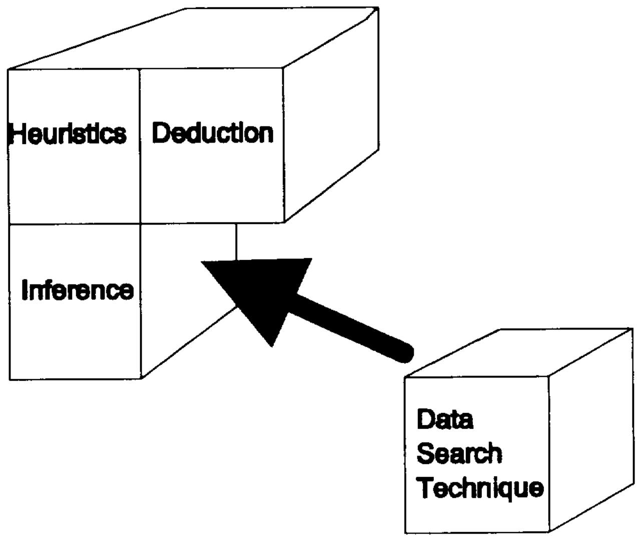


Figure 2. Addition of new capabilities to a modular design

Graphical User Interface

A point and click graphical user interface would be used to communicate a record of its actions, conditions encountered (sensor inputs or experiment signals), and health and status, or to provide human intervention. This will allow a user to configure the graphical user interface to display information in any format a user selects. The graphical user interface will support dynamic representations (widgets) or textual display of data. The user interface will display human readable information when displaying actions taken by the system (i.e., rule 3 invoked based on conditions "r" that utilized script "4.1" with the following actions.....).

Inference Engine

The MAIS inference engine will support replanning of experiment activities (adaptive planning), problem deduction (inference) and assertion based on facts (beliefs) and their relationship to rules. The inference engine will use fact, concrete and abstract representation to develop expressions for each fact, denoting a part of the fact base (rules) that the system believes. Contradictions in facts (beliefs) based on an application of rules (conflicting states) will require human intervention.

Rules

Rules are developed or modified to support a particular experiment. These rules may or may not be different for each experiment and will control the use of scripts that govern actions taken by the tool in response to sensor inputs or signals from the experiment. Each experiment can benefit from the rules and the associated scripts developed for previous experiments.

Scripts

Scripts control actions taken in response to a predefined set of conditions as stated in the rules developed for each experiment. Scripts are a non-inference technique and similar to flight software fault protection and error recovery systems.

Cognitive Replanning

Experiments that operate by utilizing detailed plans, sequential steps, (not to be confused with programs) that detect a departure from these steps require the ability to replan. Replanning will be accomplished by reordering/ deleting steps as required. Replanning will require the use of inference techniques and rules to determine a the least cost path of action to take in accomplishing the replanning objectives (see Figure 3).

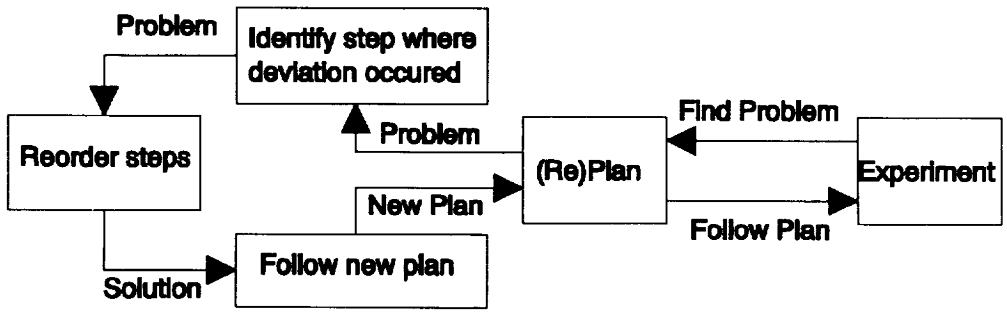


Figure 3. Replanning Model

Summary

Creation of the MAIS, a modular, reusable system that can support different types of experiments over the life of the Space Station or Spacelab program will:

1. Reduce cost
2. Allow the capture of experience from each previous experiment
3. Be reusable
4. Reconfigurable, even on orbit, to support a dynamic environment
5. Promote the development of "spin-offs" to industry
6. Reduce ground support personnel costs and free up crew time on orbit
7. Be smaller, cheaper and faster than any other system in use today

This tool can be developed quickly and could be ready to support experiments on Space Station starting at human tended capability.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE Mar/94	3. REPORT TYPE AND DATES COVERED Conference Publication

4. TITLE AND SUBTITLE AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service,and Space	5. FUNDING NUMBERS
6. AUTHOR(S) Jon D. Erickson, Editor	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lyndon B. Johnson Space Center Houston, Texas 77058	8. PERFORMING ORGANIZATION REPORT NUMBERS S-754
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) American Institute of Aeronautics & Astronautics Washington, D.C. 20073 National Aeronautics and Space Admistration Washington, D.C. 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER CP-3251
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT unclassified/unlimited Available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD, 21090; (301) 621-0390. Subject Category: 63	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The AIAA/NASA Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94) was originally proposed because of the strong belief that America's problems of global economic competitiveness and job creation and preservation can partly be solved by the use of intelligent robotics, which are also required for human space exploration missions. It was also recognized that in the applications-driven approach there are a far greater set of common problems and solution approaches in field, factory, service, and space applications to be leveraged for time and cost savings than the differences in details would lead one to believe. This insight couple with a sense of national urgency made a continuing series of conferences to share the details of the common problems and solutions across these different fields not only a natural step, but a necessary one. Further, it was recognized that a strong focusing effort is needed to move from recent factory-based technology into robotic systems with sufficient intelligence, reliability, safety, flexibility, and human/machine interoperability to meet the rigorous demands of these fields, the scope of which is beyond the capability of any one area.

The papers in these proceedings are evidence that users in each field, manufacturers and integrators, and technology developers are rapidly increasing their understanding of integrating robotic systems on Earth and in space to accomplish economically important tasks requiring mobility and manipulation. The 21 sessions of technical papers in 7 tracks plus 2 plenary sessions cover just the tip of this major progress, but reveal its presence nonetheless.

14. SUBJECT TERMS robotics, robots, computers, computer programming, artificial intelligence			15. NUMBER OF PAGES 923 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std. 239-18
298-102