NASA Conference Publication 3127, Vol. I

Fifth Annual Workshop on Space Operations Applications and Research (SOAR '91)

Proceedings of a workshop held in
Houston, Texas
July 9–11, 1991

NASA

NASA Conference Publication 3127, Vol. I

Fifth Annual Workshop on Space Operations Applications and Research (SOAR '91)

Kumar Krishen, *Editor*
NASA Lyndon B. Johnson Space Center
Houston, Texas

Proceedings of a workshop sponsored by the
National Aeronautics and Space Administration,
Washington, D.C., the U.S. Air Force, Washington, D.C.,
and cosponsored by the University of Houston-Clear Lake,
Houston, Texas, and held at
Lyndon B. Johnson Space Center
Houston, Texas
July 9–11, 1991

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1992

INTRODUCTION

Kumar Krishen, Ph.D.

Operations[1], as the term applies to the Nation's civil defense and space programs, constitutes a broad spectrum of activities and associated facilities that enable the conduct of a program or a mission to achieve the desired goals or objectives. They include Earth-based, in-flight, in-space, and planetary surface-based operations. Mission plans, schedules, and logistics are integral parts of operations. With regard to the conduct of the Space Shuttle Program, NASA has become increasingly conscious of the need for improving operations efficiency. The goal of operations efficiency efforts is to provide systems, services, and the infrastructure to enable safe operations at a substantially reduced cost. The operations Work Breakdown Structure typically includes space automation and robotics, training systems, in-space operations, ground operations, and associated information and communications infrastructure. In particular, recent advances in software technology and knowledge engineering are deemed crucial in providing revolutionary capabilities for operations associated with the space programs.

The National Aeronautics and Space Administration (NASA) and the U.S. Air Force have recognized the need for continued interaction in the area of space operations technology and formed the Space Operations Technology Subcommittee (SOTS) under the NASA/Air Force Space Technology Interdependency Group (STIG). The membership of the SOTS is listed in Table I.

The goals of SOTS include the following:

- Interchange technical and programmatic information related to space operations
- Share lessons learned
- Identify areas of common/mutual interest
- Encourage interdependent programs

In the past 6 years, the SOTS has identified research and applications areas with significant potential for Air Force and NASA programs. These areas are intelligent systems, automation and robotics, life sciences, environmental interactions, and human factors.

The focus of the SOTS is on the research and technology areas, which have applications to both NASA and the Air Force. Thus, the coordination and joint pursuit in these areas of overlap (see figure 1) would provide savings for both agencies.

In addition to program reviews, meetings, and written communication, the SOTS conducts the Space Operations, Applications and Research (SOAR) symposium and exhibition annually. This symposium and exhibition has become an invaluable tool to review the progress made in existing joint programs and to identify new areas for joint or collaborative efforts. Table II presents the program overview and personnel responsible for SOAR '91.

This document contains papers presented at the Space Operations, Applications and Research Symposium, hosted by the NASA Johnson Space Center (JSC) and held at JSC in Houston, Texas, on July 9 - 11, 1991. More than 110 papers were presented at the Symposium, sponsored by the U.S. Air Force Phillips Laboratory, the University of Houston-Clear Lake, and NASA JSC. The technical

[1]"Advanced Technologies for NASA Space Programs," K. Krishen, Proceedings of the 10th Annual International Space Development Conference, San Antonio, Texas, May 17-22, 1991.

areas covered were Intelligent Systems, Automation and Robotics, Human Factors and Life Sciences, and Environmental Interactions. The U.S. Air Force and NASA programmatic overviews and panel discussions were also held in each technical area. A keynote session chaired by Dr. Aaron Cohen, Director of the Johnson Space Center, was organized to provide Agencywide perspective of technology programs for both the Air Force and NASA. The keynote addresses of Maj. Gen. Robert Rankine, Jr., and Mr. Arnold Aldrich, who serve as the co-chairmen for the STIG, are included in this document. These proceedings, along with the comments and suggestions made by the panelists and keynote speakers, will be used in assessing the progress made in joint USAF/NASA projects and activities. Furthermore, future collaborative/joint programs will also be identified. The SOAR '91 Symposium and Exhibition is the responsibility of the Space Operations Technology Subcommittee (SOTS) of the USAF/NASA Space Technology Interdependency Group (STIG). The Symposium proceedings include papers covering various disciplines presented by experts from NASA, the Air Force, universities and industry.

TABLE I. MEMBERSHIP OF SOTS

Phillips Laboratory
Melvin Rogers, Co-chairman
Capt. Jim Skinner
Lt. Col. Gale Nelson

Armstrong Laboratory
Col. Donald Spoon
Capt. Ron Julian
Dr. Samuel G. Schiflett

SDIO/ES
Mr. Richard Iliff

AFOSR/NM
Dr. Abe Waksman

Wright Laboratory
Capt. Mike Wellman

AFOSRL/NL
Dr. John Tangney

NASA Headquarters
Mr. Mark Gersh
Mr. Geoff Giffin

Marshall Space Flight Center
Mr. E.C. Smith

Kennedy Space Center
Mr. Tom Davis

Johnson Space Center
Dr. Kumar Krishen, Co-chairman
Mr. Robert Savely
Dr. Howard Schneider

Jet Propulsion Laboratory
Mr. Wayne Schober

Langley Research Center
Mr. Jack Pennington

Ames Research Center
Mr. Allen Fernquist
Dr. Michael Shafto
Dr. Mary Connors

Lewis Research Center
Dr. Dale Ferguson

X101325M

TABLE II. PROGRAM AND RESPONSIBLE PERSONNEL

Program

SOAR '91 will include USAF and NASA programmatic overviews, panel sessions, exhibits, and technical papers in the following areas:

- Intelligent Systems
- Automation and Robotics
- Life Sciences
- Human Factors
- Environmental Interactions

Technical Sessions and Exhibit Hours

Tuesday, July 9	8:00 am - 7:00 pm
Wednesday, July 10	8:00 am - 7:00 pm
Thursday, July 11	8:00 am - Noon

Welcome/Opening Addresses (July 9, 8:30 - 9:30)

NASA/Air Force
Mr. Geoff Giffin - NASA Headquarters
Dr. Allan Schell - Air Force Systems Command

Panel Discussion (July 9, 3:30 - 5:00)
Technology Requirements

Moderator: Dr. Kumar Krishen - NASA/JSC
Panelists: Geoff Giffin, Mr. Peter Ahlf - NASA/HQ, Mr. James Romero, Col. Ray Barker/USAF

Keynote Session (July 10, 6:30 - 9:30)

Master of Ceremonies: Dr. Aaron Cohen - NASA/JSC
Keynote Speakers: Major General Robert Rankine, Jr. - Air Force Systems Command and Mr. Arnold Aldrich - NASA/HQ

Exhibitors

Computer Sciences Corp.	McDonnell Douglas
Deneb Robotics	NASA/Ames Research Center
Digital Equipment	NASA/Johnson Space Center
EXOS	Oracle Corp.
NASA/Goddard Space Flight Center	Rice University
Grumman Corp.	Rockwell-Downey
Hewlett Packard	Silicon Graphics
IBM Federal Sector	Space Industries
IntelliCorp	Template Graphics
KRUG Life Sciences	Togai InfraLogic
Lockheed	University of Lowell
LinCom Corp.	USAF

Symposium Coordinators

Symposium General Chair:	• Dr. Kumar Krishen, NASA/JSC
Assistant General Chair:	• Mr. Mel Rogers, Phillips Laboratory
Technical Coordinators:	• Capt. Jim Skinner, Phillips Laboratory • Mr. Robert T. Savely, NASA/JSC
Administrative Co-Chairs:	• Ms. Carla Armstrong, Barrios Technology, Inc. • Dr. Glenn Freedman, University of Houston - Clear Lake
Exhibit Co-Chairs:	• Mr. Charles Pittman, NASA/JSC • Mr. Ellis Henry, I-NET, Inc. • Ms. Bette Benson, University of Houston - Clear Lake

Technical Area Coordinators

	NASA	USAF
Intelligent Systems	Mr. Mark Gersh NASA/HQ	Capt. Jim Skinner Phillips Laboratory
Automation and Robotics	Mr. Jack Pennington NASA/LaRC	Capt. Ron Julian Armstrong Laboratory
Life Sciences	Dr. Howard Schneider NSA/JSC	Dr. Samuel G. Schiflett Armstrong Laboratory
Human Factors	Ms. Mary Connors NASA/ARC	Col. Donald Spoon Armstrong Laboratory
Environmental Interactions	Dr. Dale Ferguson NASA/LeRC	Lt. Col. Gale Nelson Phillips Laboratory

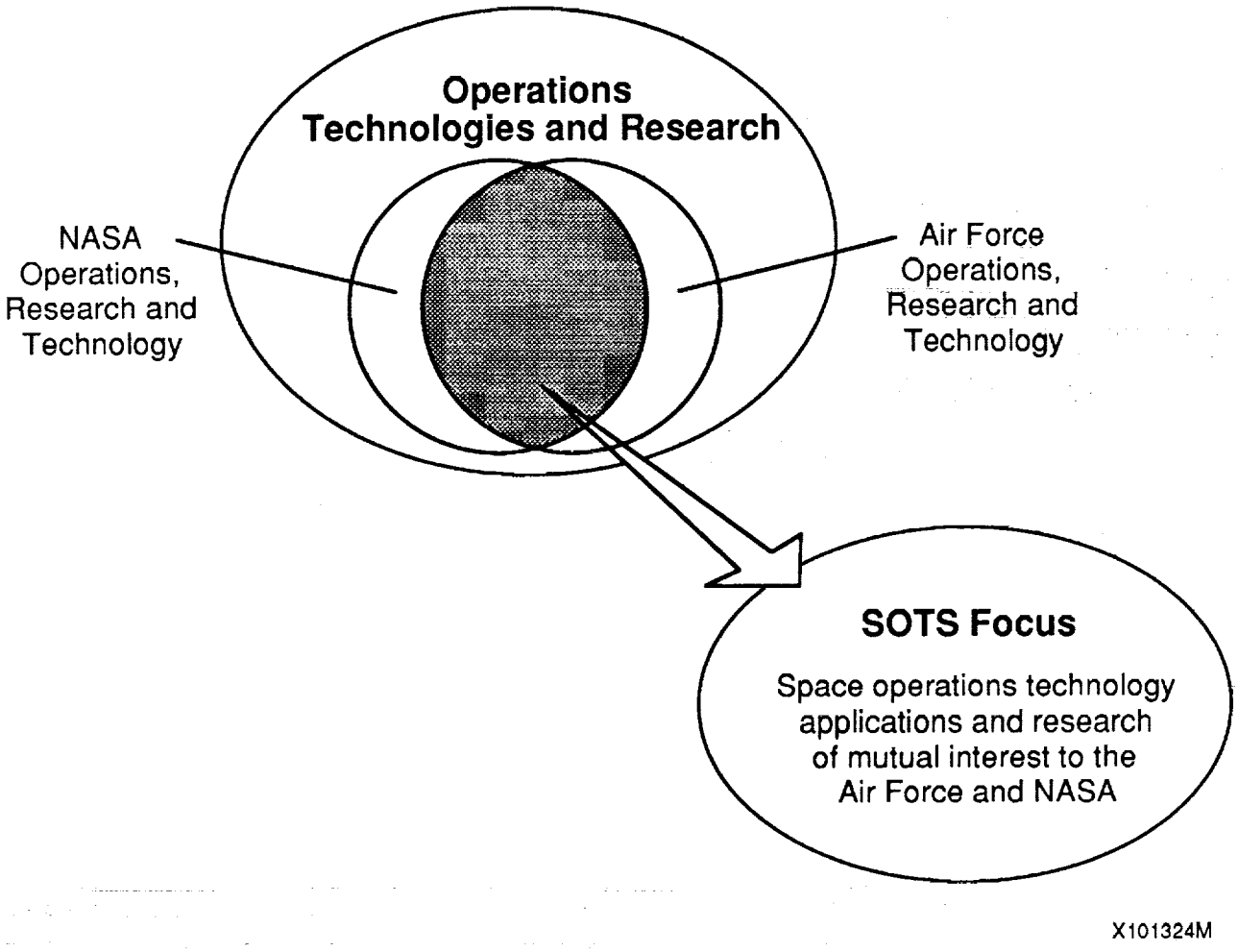


Figure 1. The SOTS Domain of Technologies

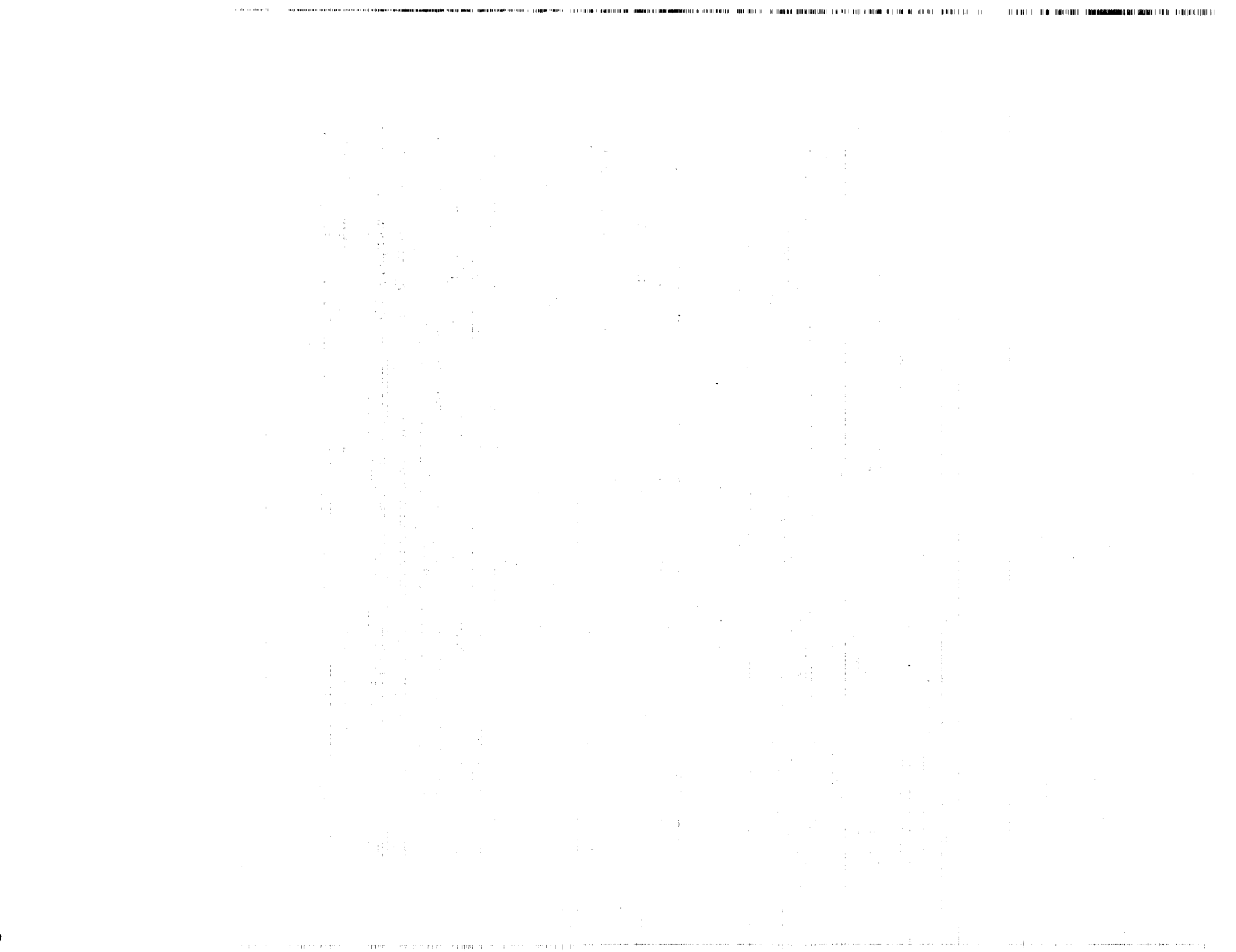
MESSAGES

General Chair,
Dr. Kumar Krishen
NASA Johnson Space Center

The goals of the Space Operations Technology Subcommittee (SOTS) of the Space Technology Interdependency Group (STIG) include interchange of technical and programmatic information, sharing of lessons learned, and the identification of interdependent programs related to space operations. The SOAR Symposium and Exposition has been, and continues to be, an excellent means to accomplish most of the SOTS goals. This is the fifth time the Air Force and NASA will host the Symposium and Exposition to evaluate progress of our ongoing efforts and identify future cooperative programs. To this end, exhibits, technical papers, panel discussions, and programmatic reviews have been planned for the technical interchange. The present fiscal climate of our nation makes it incumbent upon us to avoid duplication and embark on cooperative and joint technology development for Air Force and NASA applications. With your participation, I believe we can achieve this coveted goal and advance the future programs of both NASA and the Air Force.

Assistant General Chair,
Mel Rogers,
Kirtland AFB

As the new Air Force Co-Chairman for the Space Operations Technology Subcommittee, I would like to welcome your participation in the 1991 SOAR Conference. The past year has been a dynamic one within the Air Force. We have seen the restructuring of the Air Force. We have seen the restructuring of the Air Force laboratories and the implementation of Project Reliance to increase cooperation between the services. As a result of the restructuring, the Phillips Laboratory has been designated as the new Air Force superlab dedicated to developing technologies for space. Our participation in past SOAR conferences has enabled Air Force and NASA program and project managers to reduce the duplication of effort and revealed new application areas for existing technologies. In addition, it has given the experts from both organizations the opportunity to exchange information about the technologies that are critical to space operations. The contacts that have resulted from these interchanges are a valuable resource that lasts year 'round. This year's workshop promises to be another excellent forum to exchange technical information and identify opportunities for joint and cooperative ventures related to space. I look forward to meeting with all of you at SOAR '91. I know that with your participation the conference will be informative and productive.



KEYNOTE ADDRESSES

Maj. General Robert R. Rankine

MAINTAINING TECHNOLOGICAL AND INDUSTRIAL SUPERIORITY

Ladies and Gentlemen, it is an honor for me to be with you this evening. I would like to begin my opening remarks by sharing with you a reminder from our late president—John F. Kennedy:

> "The most powerful single force in the world today is neither communism nor capitalism; neither the H-Bomb nor the guided missile—it is man's eternal desire to be free."

Freedom is the greatest gift a nation can offer its people. In a world plagued by constant turmoil and by war, in a world where most of the population is ruled by dictatorship, America's technological and industrial superiority has sustained the beacon of freedom throughout the world, providing all of us hope for a better future.

America won the cold war with technology. We won decisively in the Kuwaiti theater. However, the overall U.S. lead in technology relative to the rest of the world has eroded over the last 20 years, and the outlook is for greater technological competition in the future. In the years ahead, the U.S. will face major challenges resulting from increased foreign competition and erosion of our technological and industrial base.

Our declining industrial competitiveness is a potential source of international instability, a threat to our industrial self-sufficiency, both military and commercial, and the key to becoming a second-rate economy. The prospects of increased international competition and the relative decline in our technological and industrial base have recently received high-level attention. For example, according to Donald Atwood, Deputy Secretary of Defense, "The deterioration of America's industrial base is one of the most pressing issues facing the Department of Defense."

To prevent the continual erosion of our industrial base, we must enhance our technology base. As funds decline, technological superiority will become the backbone of deterrence and stability in the 1990s. Foreign competition, the relative decline in our industrial base, and the combined problems of technology diffusion and weapons proliferation will all make it that much harder to keep our technical edge. We must plan carefully, finding ways to do our business better, and advance technologies that will ensure America's future security and economic growth. Space is one of the major technological areas that will accomplish this goal and provide us with the greatest future flexibility.

The importance of space for national defense, technological, and industrial growth has grown—and will continue to grow. If this country is to maintain its historic leadership role, our commitment and dedication to space activities must continue to expand. In fact, as we have seen from "Operation Just Cause" and from "Desert Storm," space systems have become an integral part of military operations. Rather prophetically, several months prior to the incidents that led up to Desert Storm, Secretary Rice remarked:

> "Space is a natural extension of the Air Force's operating medium. In an unstable world with refocused threats, space offers stability and control...to be a 21st century superpower, the United States needs the ability to help friends and quell enemies within hours. Only with aerospace forces can you concentrate and reconcentrate power that quickly."

These words underscore the power and value of space technology in securing a strong and safe future for America. But space is not just a military resource, it is an invaluable resource to commerce and it is critical to our economic future. America's projected investment in space for 1991 is approximately $30 billion. NASA and the Department of Defense will spend nearly $27 billion, and industry will spend almost $3.6 billion purchasing and launching satellites. The utilization of space as a resource is realistically still in its infancy, yet the benefits derived from this resource profoundly affect the world around us.

Space technology and development will become as critical to our country's future as the industrial revolution was critical to our country's growth in the 18th century. Future growth of American industry, technology, and political strength are inextricably tied to space. America is tied to space because of its importance to both the national security and civil space sectors—it helps maintain the peace and enhances the life of every American living today. From communications to entertainment, surveillance to navigation, weather forecasting to storm warnings, science to space station, and the list goes on and on, space is central to American development today and tomorrow.

The problem that confronts all of us is "How do we sustain progress in defense technology—and, in particular, in defense space technology—in a time of declining defense budgets?" One way is to streamline in order to get more out of our limited funds. To achieve this goal, Mr. Richard Cheney, our Secretary of Defense, instituted the Defense Management Review process. Subsequently, Deputy Secretary of Defense Atwood challenged the Military Services by issuing a draft Defense Management Report Decision. This draft report decision, DMRD 922, focused on the Consolidation of Research and Development Laboratories and Test and Engineering Facilities of the military services. DMRD 922 became the catalyst that sparked formal discussions on ways to enhance cooperation and increase reliance among Service programs and facilities for research and testing.

Responding to the challenges of the DMRD 922, the Services each initiated its own laboratory consolidation proposal and collectively, the Services initiated Project Reliance to place more dependence on one another's capabilities. In response to the DMRD 922, the Air Force has taken two major steps to streamline its laboratory management structure and improve the quality and relevance of Air Force science and technology programs. The first step consolidated our scientific and technology resources at 14 different laboratories and research centers into four major laboratories. This realignment facilitates greater interaction between scientists and engineers who must develop the technology for the interdisciplinary Air Force weapon systems of the future. Our "super" laboratories will also reduce management overhead. At the same time, simpler laboratory organization will clarify and focus our missions and create more opportunities for Tri-Service Reliance and external recognition.

The four major laboratories were organized around the applied technologies that serve their parent product division. Armstrong Laboratory at Brooks AFB in Texas supports "human systems; Rome Laboratory at Griffiss AFB in New York supports "command, control, communications, and intelligence"; Wright Laboratory at Wright-Patterson AFB in Ohio supports "air vehicles and armament"; and the Phillips Laboratory at Kirtland AFB in New Mexico concentrates on space, missiles, and directed energy technology. Each of the four "super" laboratories have also been assigned certain corporate research responsibilities, such as materials research at Wright Laboratory and geophysics research at Phillips Laboratory. This reorganization will enhance our technology, help us meet our user's needs, and improve our acquisition capabilities through the increased synergy and teamwork that will bring our people, processes, laboratories, and their SPO customers closer together. As an end product, we look forward to greater combat capability for the Air Force and the Nation.

The second major step we've taken established an investment planning process to ensure the quality and relevance of our Science and Technology programs. Our Air Force acquisition executive, Assistant Secretary Jack Welch, requested the Air Force Scientific Advisory Board to review all Air Force Science and Technology programs, and we have further asked the Board to explicitly evaluate

and score those programs based on their technical quality. Similarly, we have asked the Air Force's operational commands to score the importance of our major technology demonstration programs. Using this feedback from the Air Force Scientific Advisory Board for technical quality and the Air Force using commands to determine the operational relevance, we will keep a balanced focus on customer needs while advancing technology innovation in new research areas.

Our vision for Air Force laboratories will ensure they remain world class organizations working in technical areas vitally important to the Nation's economy and national defense. But we must also enhance cooperation and increase reliance among all government research organizations if we are to maximize our available resources to strengthen our national technological capabilities.

During the summer of 1990, the three Services presented a coordinated proposal to Mr. Atwood that outlined several approaches to laboratory consolidation and inter-Service Reliance. Once Mr. Atwood conceptually approved of the Tri-Service proposal, the Army, Navy, and Air Force implemented the Tri-Service Project Reliance process to identify ways of achieving inter-Service science and technology consolidation.

By November, Mr. Atwood signed the final version of DMRD 922 and directed the Acquisition Undersecretary to provide a plan outlining management actions to implement DMRD 922. Subsequently, in December 1990, the Joint Logistics Commanders institutionalized the Project Reliance philosophy and process through the Joint Directors of Laboratories. The charter of the Joint Directors of Laboratories has been broadened in scope to conduct inter-Service Reliance analysis and joint technology planning on a continuing basis and to implement the results.

Before Project Reliance, the primary method of interaction among the Services was coordination and exchange of information at the working level. The Tri-Service Reliance activity is a comprehensive effort to identify means for increasing inter-Service Reliance in Science and Technology programs and to give the resulting joint programs visibility at the Director of Defense Research and Engineering level. This effort has yielded substantial positive changes for achieving increased efficiencies through collocations, consolidations, and joint Service S&T programs. Formalized Service agreements have been established for each of the technologies considered that were not Service unique. As a result of these agreements,

- There are 71 technology sub-areas where the Services will jointly plan for work to be conducted at separate Service locations.

- There are 105 technology sub-areas where work will be collocated to various single-Service sites for program execution.

- There are 4 major technology areas and 6 technology sub-areas where programs will be consolidated and carried out at a single location under a lead service for management.

The four objectives of continuing oversight of Tri-Service science and technology by the Joint Directors of Laboratories are:

1. Structuring an integrated DoD S&T program in selected technology areas through joint program planning and execution;

2. Establishing technology panels to plan and oversee execution of cooperative Service S&T programs;

3. Collocating S&T efforts to achieve critical mass and avoid unwarranted overlap and eliminate duplication among Service programs; and

4. Conducting annual joint and corporate Science and Technology reviews for the Director of Defense Research and Engineering and his staff

These activities will enable all Services to increase their efficiencies in areas of common interest without severing the close ties to the end users of the technology and will free up resources, which each Service can reinvest in critical, higher-priority technology areas. These changes will reduce the perception of program fragmentation and duplication, promote an integrated "joint" Tri-Service program, and enhance the overall quality of DoD's S&T program. through this improved strategic planning process, we will have greater control over our technological future.

We now wish to extend this collaborative spirit to other Federal agencies that invest in science and technology that is relevant to defense needs. By expanding the membership of the previously established NASA/Air Force space Technology Interdependency Group to include the Army and the Navy, we have created a forum for Tri-Service cooperation with NASA in space technology. That is why we are currently holding this combined Space Technology Interdependency Group and Joint Directors of Laboratories meeting, to couple previously successful Tri-Service cooperation with previously successful Air Force/NASA cooperation.

In the past year, the Services have made remarkable progress in establishing a new era of Science and Technology cooperation. IntraService laboratory consolidation reduced our management overhead, helped us focus our efforts, and increased our Tri-Service Reliance. The changes within the Army, Navy, and Air Force Laboratories have greatly strengthened our technical capabilities. Tri-Service Science and Technology cooperation has never been better. We wish to expand this Science and Technology cooperation to other federal agencies beginning with NASA. Working together as a team, we should be capable of maintaining our world leadership role in space technology. In fact, by working as a team, we should be able to far exceed the objective of merely maintaining a world leadership role; we should be able to lead our nation into a new golden era of space technology and applications.

Arnold D. Aldrich
NASA Associate Administrator
Office of Aeronautics
Exploration and Technology

COLUMBUS, VISIONARIES, AND BOLD ACTION

The past is beyond our ability to change, but the future is ours to make, as we see fit. Christopher Columbus, in his time, was in no way able to visualize the future on a five-hundred year time horizon, and we who will soon celebrate his far-reaching achievement are not much better prepared to visualize our future on such a long-range time scale.

Columbus and his small band of individuals did, however, in their courageous pursuit of a vision of new routes to the Orient, instead open up new avenues to economic prosperity and thrust the world forward into the current era. With the daring to press beyond the constraints of their time, they activated a process that led to acquiring new levels of knowledge about the world in which we live.

That same spirit was alive and well a hundred years later when the colonization of North America began, and it was even stronger at the end of the 18th Century when the United States was formed and the task of building civilization across this continent moved forward in earnest. Although it has been many years since the western frontier vanished, that heritage, the heritage of the explorer and the builder, endures within America today.

Nowhere today does that spirit burn more brightly, nor is the legacy of the explorer and the builder embodied more profoundly than in the United States space program. And it is within this space program, with the men and women of NASA, the Department of Defense, the Department of Energy, the aerospace industry, and the academic research community, that we, as a nation, have the opportunity to adopt a vision and initiate bold action for the future. If the United States is to prosper in the next century, we must have a guiding perception of the future as it could be, and work today to build towards those goals.

For the present, our nation's space program is leading the world, but in the next century, with the pace of current international competition, it could well be relegated to second class, or even to a lesser status. George Santayanna said that those who fail to learn from the mistakes of history are doomed to repeat them. Tragically, history shows that those societies that fail to aspire to a challenging vision of the future will realize a self-fulfilling prophecy of diminishing accomplishments.

Twenty-five years from now, in the year 2016, we could be achieving exciting, inspiring successes in space. The frontiers of space science, the return to the Moon, the exploration of Mars, and understanding Earth itself all challenge us. But what, in fact, will we actually accomplish over this timeframe?

I would like to remind you of the motion picture "2001 - A Space Odyssey." Created in the 1960s, Arthur C. Clarke's vision of a possible future in space stands largely unfulfilled. That vision was shared by towering figures of the early U.S. space program such as Webb, Gilruth, Von Braun, and the other leaders who guided the Agency through the bold space adventures of the 1960s, 1970s and the early 1980s.

The question we must ask ourselves now, 30 years later, is whether we still have the vision of a nation of explorers and builders? If we envision great accomplishments in the coming century, what steps must we take in this decade to make those goals achievable? Conceiving or embracing inspired visions is not enough; we must also act with conviction.

I believe a central, crucial path of action is to build on our outstanding capabilities in technological research and innovation. We must make a renewed commitment to the development of advanced space technologies. To expand our frontiers in space, we must challenge established technologies and explore new approaches to our space endeavors. We must find better ways if we are to achieve greater heights.

It has been widely demonstrated that the space program is both *driven by* and also is a proven *driver of* technology. Just as the explorations of past centuries have resulted in tremendous economic benefits, so too can our space program of the next century provide equally vital returns.

However, we must also have the courage to acknowledge our present difficulties, and we must have the determination to move effectively to overcome them. Despite our past accomplishments, today the art of space engineering is relatively new and immature, comparable perhaps to the state of engineering in aeronautics in the 1930s or perhaps even the 1920s. As a consequence, we have the need for a broad range of technological advancements that must be vigorously pursued.

Many national leaders have articulated their concerns over the current status and outlook for the United States space program and have emphasized the need for stronger investments in research and technology to enhance our competitiveness. Most recently, the report by the Advisory Committee on the Future of the U.S. Space Program, otherwise known as the "Augustine Committee," stated that NASA should provide a stable share of its budget for space technology and fund it at two or three times the current level of investment. They also recommended that NASA formulate an agencywide technology plan and subject it to a broad review by experts external to the Agency.

We are currently deeply immersed in formulating a comprehensive "Integrated Technology Plan" (ITP) which is responsive to these recommendations. And we have just completed a week-long review of this plan by specialists from industry, academia, and other agencies. However, we recognize that in the world of budget realities, it may be some years before NASA can fully support an expanded space technology program at the level recommended by the Augustine Report.

Critical, technology challenges are highlighted in the ITP, technologies that have been emphasized by the Augustine and Synthesis reviews, as well as by our own 90-Day Study. Some of the major challenges are the development of technologies for

- Space nuclear power and propulsion,
- Heavy lift launchers and orbital transfer vehicles,
- Information collection, processing, and visualization for data quantities equivalent to multiples of the Library of Congress, and
- A myriad of human support systems, including regenerative life support and EVA systems for both in-space and surface operations.

As we are all well aware, space technology budgets are under pressure, not only in the NASA civil space program but also in the Department of Defense, the Department of Energy, and in other agencies as well. Shortfalls in these areas, in turn, adversely affect industry's Independent Research and Development (IR&D) budgets and university research grants and contracts. Thus, this becomes, more than ever, a time to leverage all national technological resources by communicating and,

wherever possible, coordinating research and technology program goals, plans, and products. It is a time for partnerships in the national interest within government and between government, industrial, and academic sectors.

The list of space program challenges and opportunities is formidable, and many of them can be effectively addressed through advances in technology. In particular, the costs of year-in and year-out space operations must be actively pursued. Launch costs are too high, and launch operations are complex and time-consuming. If we are to achieve our goals in science and in exploration, we must have both low cost, reliable, cargo transportation to orbit and assured access to space for human operations. Together, steps to control operations costs and adequate near-term investments in advanced technology and transportation systems can provide the foundation for the space program of the future.

The DoD vision of achieving military technological superiority, so aptly demonstrated in the bold U.S. response in Desert Storm, is symbolic of what this nation is capable of accomplishing. Today, the civil space program must have the vision to see beyond the constraints of our time, to envision how we can open new routes to national success and economic prosperity and to help move this nation to the forefront of a new era. There's so much out there that needs doing, so many science and exploration goals yet to be achieved, so much that is still unknown. That is what is exciting. We must anticipate what the future could be, and more importantly, we must resolutely take the bold steps in technology necessary to ensure that that future comes to pass.

The occasion of this banquet is certainly a tribute to the potential for cooperation in space technology activities between many contributing sectors of our industry. Tomorrow and Friday the Space Technology Interdependency Group, STIG, takes a visionary step toward becoming a broader federal forum for technology cooperation with the addition of U.S. Army and U.S. Navy membership, and there will also be informal participation by representatives from DARPA, SDIO, and DOE.

As you have heard from General Rankine, the tri-services technology panels, under the Joint Directors of Laboratories, have met here today with a focus on space activity. And the week's underpinning event is the Space Operations, Applications, and Research (SOAR) conference. This activity, conceived by the STIG's Space Operations Committee has, for the past five years, created a cooperative forum for NASA and the Air Force to reach out to industry and academia.

I congratulate all of you, the leadership and participants in this "triple-header" demonstration of national technological cohesiveness, and I believe these activities will greatly assist in building the cooperative relationships so essential to the realization of each of our organizational visions. These activities serve as a model for the leveraging of total U.S. capabilities toward the goal of continuing space program preeminence.

CONTENTS

SECTION II: AUTOMATION AND ROBOTICS

Session A1: AUTOMATION AND ROBOTICS PROGRAMMATIC SESSION
Session Chair:

Session A2: SENSING AND DISPLAY
Session Chair: Prof. Neil Duffie

Session A3: ERROR DETECTION AND CONTROL
Session Chair: Joe Herndon

Session A4: MANNED PERFORMANCE
Session Chair: Dr. Mark Stuart

SECTION III: HUMAN FACTORS AND LIFE SCIENCES

Session H1: MEASUREMENTS, TOOLS, AND ANALYSIS - I
Session Chair: Dr. Mary Connors

Session H2: SPACE PERCEPTION AND PHYSIOLOGY - I
Session Chair: Dr. Samuel Schiflett

Session H3: MEASUREMENTS, TOOLS, AND ANALYSIS - II
Session Chair: Col. Donald Spoon

Session H4: HUMAN MACHINE INTERACTIONS
Session Chair: Dr. Arthur Beller

SECTION I

INTELLIGENT SYSTEMS

Session I1: AUTONOMY

Session Chair: Maj. Carl Lizza

'From Pilot's Associate to Satellite Controller's Associate"

Major David L. Neyland
Advanced Systems Technology Office
Defense Advanced Research Projects Agency

Major Carl Lizza
Cockpit Integration Directorate
Wright Laboratories, USAF

Mr. Philip A. Merkel
BDM International, Inc.

Introduction

Associate Technology is an emerging engineering discipline wherein intelligent automation can significantly augment the performance of man-machine systems. An associate system is one that monitors operator activity and adapts its operational behavior accordingly. Associate technology is most effectively applied when mapped into management of the human-machine interface and display-control loop in typical manned systems. Fundamental to an associate system is an embedded operator model with which operator activity is compared. Inferences can be made about the state of the operator, and the system can respond in different ways dependent upon both the external situation and the local requirements of the operator. These technologies are being developed under the auspices of the DARPA Pilot's Associate (PA) Program.

One logical exploitation and application of associate technology is for intelligent command and control of remote assets, particularly for satellite systems. Such a system might be a satellite controller's associate, comprised of a community of coupled software processes embedded in the satellite control facilities computers. These processes would digest external world knowledge through sensed data fusion and correlation and generate inferences based on models of expected behavior for the real world. The processes would generate action plans for the operator to satisfy his goals and constraints as he continuously interacts with the external and internal environment. The processes would abide by the paradigm that the operator is always in charge, and that the data must always go through. The processes would act as decision aiding tools, incorporating display management, situation assessment and mission planning. These capabilities are all incremental evolutions of the techniques developed under the Pilot's Associate program.

This paper addresses the potential for application of associate technology into the arena of intelligent command and control of satellite systems, from diagnosis of onboard and ongound of satellite systems fault conditions, to execution of nominal satellite control functions. Rather than specifying a specific solution, this paper draws parallels between the Pilot's Associate concept and the domain of satellite control.

Program Objectives

The forerunning program, the Pilot's Associate, started with the nearly irreducible minimum crew size of one pilot in a fighter aircraft. The objective of the PA was to improve the performance of the less experienced pilots by the addition of a "virtual crew member" which could address some of the information processing limitations of the human. In part, this can be described by the threshold of performance expectation across the spectrum of operator or pilot experience. These limitations, as described by the McDonnell Aircraft PA team [Reference 1], included working memory, speed of cognitive operations, reliable retrieval of information, accuracy of numerical operations, and projection of position in time and space. These are all areas where computers excel in aiding human performance. To surmount these limitations, the PA was designed to perform internal and external situation assessment, long-range and reactive planning based on those assessments, and intelligent communication with the pilot. These functions serve to improve the situation awareness and decision making of all

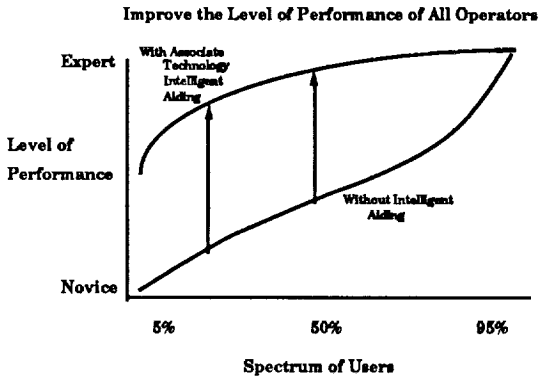


Figure 1--Human Performance Goals

operators, with respectively more payoff for the less experienced pilots. The top level goal is to improve the performance of the entire population and thereby obtain aggregate force effectiveness improvements (Figure 1--Human Performance Goals).

A satellite controller's associate program might have a much different objective; that being to increase effectiveness with dwindling manpower resources. It is always interesting to the uninitiated to observe the complexity and manpower intensity for monitoring, maintaining, and operating satellite systems. Currently, over 4000 personnel are necessary to man the Air Force Satellite Control Network to operate approximately 80 satellites. Predictions suggest that 135 satellites will be in place by the year 2000, and 150 will be in orbit by 2015 -- not including any added by the Strategic Defense Initiative. The number of complex satellite systems doubles, yet the number of personnel available to operate them is likely to remain constant or decrease. The use of machine intelligence to augment the abilities of satellite controllers can improve their operational effectiveness.

The incorporation of decision aiding systems can be described at three levels. Automation, at the lowest level of machine control, removes human interaction except for certain specific activities, but relies entirely on a'prior knowledge and planning to adequately respond to a rigorously predefined environment (Figure 2 -- Human-Machine Interaction). Assistant systems provide sophisticated computer mechanization to allow a human operator to manipulate and manage complex information and actions upon specific operator request along the lines of traditional expert systems. Associate systems encompass the virtues of automated and assistant systems, but

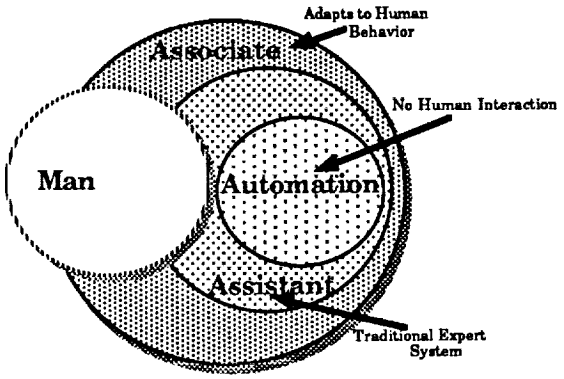


Figure 2 -- Human-Machine Interaction

add a level of internal and external environment awareness and monitoring that enables machine adaptation to both environmental constraints and human behavior in-the-loop.

Automation of routine spacecraft operations, stationkeeping, commanding and maintenance tasks has long been considered and implemented to varying degrees as a method of enhancing controller efficiency and effectiveness. In general, however, discrete task automation by itself may be insufficient and indeed undesirable, with its potential to produce unintended scripted results counter to the critical nature of many tasks.

Ten years ago four major policy goals of automation were identified to improve performance of space missions [Reference 2]. Those were:

1) Ground interaction reduction,
2) Spacecraft integrity maintenance,
3) Autonomous features transparency,
4) Onboard resource management.

To achieve these goals, levels of autonomy were identified from 0 through 10, ranging from an open-loop, non-redundant, ground-controlled onboard system, to a fully autonomous system capable of internal reorganization and dynamic task deduction based upon unknown and unanticipated changes in the environment [Reference 3]. The trend over the past ten years has been to varying degrees an increase in the autonomy of the space segment and reduction of dependency on the ground segment. A low risk approach to reduce the reliance on manned ground control systems could be the incorporation of intelligent decision aiding. Many of the routine tasks eventually to be delegated to satellite ownership can be implemented and validated within an intelligent ground control system.

Within the bounds of the autonomy spectrum, it is possible to visualize a continuum of transfer of workload from the human operators into the domain of machine control, *within the ground control facility*. With increasing satellite autonomy, respective operator intervention, action and workload logically decrease. This is a workload baseline. From that

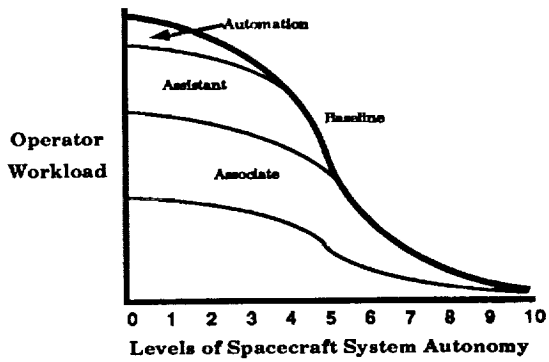


Figure 3 -- Workload Reductions

baseline, automation of ground operations tasks further reduces workload, but only for those functions not already within the autonomy domain (Figure 3 -- Workload Reductions). Assistant systems can provide additional workload reduction, but again only partially throughout the

systems or automatic transmission of command string sequences. The associate concept adds the sensed and observed feedback of operator performance to tailor the output of the associate to support the best estimate of the operator's needs at the time.

The fundamental interaction between the operator and the associate is through the approval of plans. The associate generates plans which tend to satisfy the pre-established goals of the operator (Figure 5 -- Planning Flow). If the operator rejects a plan,

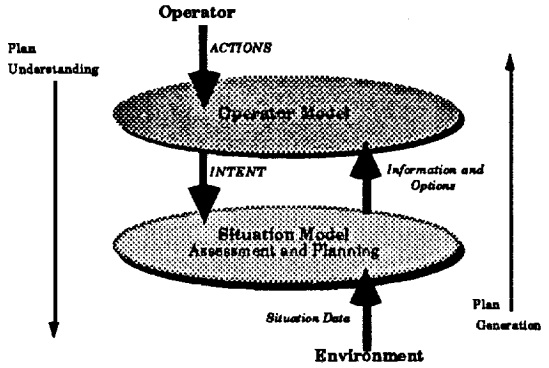


Figure 5 -- Planning Flow

for whatever reason, the next most reasonable plan is proposed. The operator can accept plans either explicitly by manual consent action or implicitly by following the course of action suggested by the plan. This is an intrinsic element of associate technology architectures. Response is governed by the philosophical construct that the operator is always in command. Paramount is the notion that the workload reduction provided by the associate system must be greater than the effort required to control the associate system. These precepts are embodied below:

1) The operator may perform any action at any time and may override any associate system proposal or action.
2) All potential associate system actions are considered only as proposed plans until the operator approves them.
3) The authority given the associate system to perform actions can be controlled by the operator.
4) The requirement that the associate system notify the operator of key mission events and associate system actions can be controlled by the operator.
5) The plans proposed by the associate system can be preselected, weighted and tailored prior to the mission.
6) The plans proposed by the associate system must accommodate explicit or implicit acceptance or rejection.
7) The associate system must always behave in a well understood, predictable manner.
8) All associate system actions must be tailorable by the operator.
9) The associate system must monitor and adapt to operator activity, not vice versa.
10) The associate system must follow the operator's lead.

Technology Transfer

The transferable technology from the DARPA PA program falls into two major categories: the software development process for integration of complex knowledge-based software systems and the inherent commonality of the underlying associate system architecture with its respective development environment.

The software development process began as the crafting of conventional expert systems. The well-known artificial intelligence shells and the foundation Lisp language were used to provide the system functionality, but these constructs suffered from lackluster real-time performance. The PA system matured into a hybrid system with a preponderance of the software better described as procedural than as declarative. Major embedded functions, such as route optimization and subsystem performance models, were used in their native form and language. As the program impetus to achieve real-time performance grew, the need to use the more efficient languages of C and C++ became

apparent. In retrospect, the artificial intelligence environment was found to very useful in defining requirements and developing functionality using rapid prototyping. As the system evolved to a more deterministic nature, conventional software development procedures become more appropriate.

The PA program is generating a substantial experience base in adapting intelligent software to standard hardware environments. The experience of the program since January 1990 has been oriented toward moving the five cooperating expert systems and the ancillary support functions from the Lisp laboratory environment to a "brassboard" avionics environment. In the case of McDonnell Aircraft, the Phase I environment consisted of TI Explorer machines linked to the avionics displays of an aircraft research flight simulator. For Lockheed, the Phase I environment consisted of Symbolics machines and a Micro-VAX driving a specially built cockpit. The Phase II Lockheed environment consists of RISC processors hosted in a VME chassis and using the VxWorks operating system. The resulting system provides data flow similar to what one would expect in a federated architecture avionics system (Figure 6 - Pilot's Associate Data Flow).

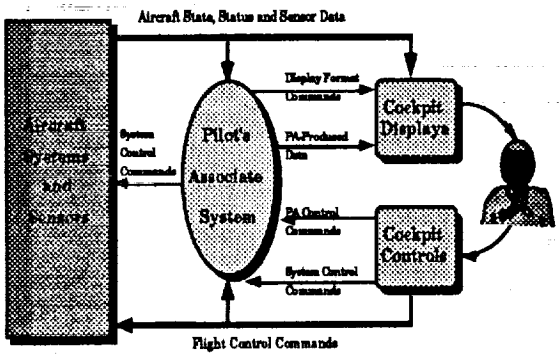


Figure 6 - Pilot's Associate Data Flow

The PA program has adapted the MIL-STD-2167A software development standard to accommodate a rapid prototyping development cycle. As a research-oriented endeavor, the program capitalized on the merits of rapid prototyping. The rapid prototyping process includes both the definition and refinement of requirements and the incremental coding, integration, and testing of functionality. This iterative process is not readily supported by the MIL-STD 2167A prescribed process. As a result, the documentation process reflects software "as built" with performance specifications evolved during development. The implication of this software management approach is that close involvement of the technical management team is essential to ensure that the directions and results match the intent and goals of the overall program.

Knowledge Engineering

The knowledge base of the Lockheed Pilot's Associate is described with a Plan-Goal graph. The top-level goals are expressed as COMPLETE-MISSION, PERFORM-MISSION-ROLE, RETURN-GOOD-AIRCRAFT, and SURVIVE. Various plans support the achievement of these goals. A similar structure might be developed for a Spacecraft Controller's Associate knowledge base. Here the top- level goals might be COMPLETE-OPERATIONAL-TASK and ASSURE-OPERATIONAL-LIFETIME. Plans which have sub-goals addressing each of the operational sub-tasks for the system would support one or the other of the top-level goals.

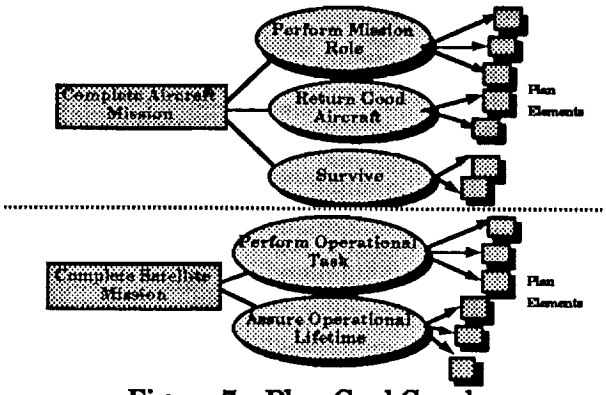


Figure 7 -- Plan Goal Graph

The organization of system operational knowledge into a plan goal graph provides a ready method for the resolution of conflicts. Where goals conflict, there will always be a higher level goal with some relative importance attached. The relative importance of conflicting goals is dependent upon the current situation and context. These elements

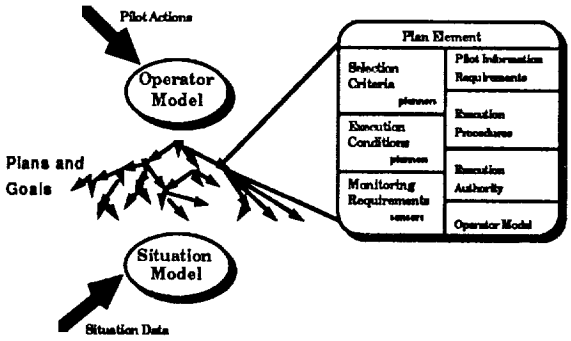


Figure 8 -- Plan Elements

are utilized within the PA as the basis for conflict resolution. Each node in the Plan Goal Graph is considered a planning object, containing sufficient detail to enable plan generation and plan understanding (Figure 8 -- Plan Elements). The same mechanism could be used to resolve the conflicts that might arise in operational control of a satellite, such as orbit adjustment conflicting with attitude control fuel usage.

The PA is designed along parallel consideration of immediate, reactive planning and long-term, mission optimization. These considerations are explicitly represented in the tactics planner and the mission planner. The plans produced by these two planners make use of system resources (for which the current state is maintained by the system status sub-system) and current situation information (for which the current state is maintained by the situation assessment sub-system). Because a satellite controllers associate would not be constrained to supporting a single operator, the planning function might be divided along the lines of operations and maintenance, rather than tactics and mission. The single pilot-vehicle interface of the PA might also be replaced by specialized intelligent interface units tailored for the specific function of each individual control position.

Associate Technology Trends

The inaugural associate program has been the DARPA Pilot's Associate program. Begun in 1986, the PA was designed as a military application within the parent Strategic Computing Program with the objective of integrating real-time, cooperating expert systems. The mechanisms for communication and cooperation between the expert systems are now well proven. Designing and building expert systems for real time operation is on track for demonstration in early 1992.

DARPA has also pursued a Submarine Operational Automation System (SOAS) which draws upon the decision aiding concepts of the Pilot's Associate. The SOAS supports tactical and mission planning with specifics

in signature management, resource management, damage control, and weapon planning. The SOAS provides this support to the skipper and battle staff of an attack submarine, addressing the issues of hierarchical planning and multi-agent reasoning.

The Army Aviation Systems Command (AVSCOM) has followed DARPA with the establishment of a Rotorcraft Pilot's Associate program. Like the DARPA-Air Force PA, the RPA program emphasizes cognitive decision aiding. AVSCOM has chosen a limited mission area for initial development and flight demonstration, that being cognitive support during a day or night, adverse weather pilotage mission. This mission can be equated to an under-the-weather, survivable, infiltration/exfiltration helicopter flight mission. This program is one year into a two-year preliminary design phase involving early determination of specific measures of effectiveness.

Conclusion

Both the design philosophy and the architecture of the Pilot's Associate system are sufficiently general to permit fairly direct application to other decision aiding environments. The packaging and throughput requirements of the fighter aircraft application were very restrictive. Achieving complex, computer-aided decision support for applications where volume, weight and power constraints are less severe provides opportunity for growth and enhancement. It would seem that the time is appropriate to consider this advanced form of operator decision aiding in the context of space mission operational requirements: a Satellite Controller's Associate.

REFERENCES

1. Banks, Sheila B., Carl S. Lizza. "Pilots Associate: A Cutting Edge Knowledge-Based System Application", Wright Research and Development Center, Wright-Patterson Air Force Base, OH, (February 1991)

2. Barry, John M., Linas Raslavicius, Thomas P. Gathmann, Joyce Kubo. Satellite Autonomy Standards for Autonomous Operations, AIAA Autonomy Subcommittee Paper, (no date)

3. Marshall, M.M. "Goals for Autonomous satellite, USAF Report SD-TR-81-72, JPL Report 7030-1, (March 31, 1981)

A Model-based Reasoning Approach to Sensor Placement for Monitorability

Steve Chien, Richard Doyle, and Luiz Homem de Mello

Artificial Intelligence Group, Jet Propulsion Laboratory, M/S 301-490,
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099

Abstract

This paper presents an approach to evaluating sensor placements to maximize monitorability of the target system while minimizing the number of sensors. The approach uses a model of the monitored system to score potential sensor placements on the basis of four monitorability criteria. The scores can then be analyzed to produce a recommended sensor set. An example from our NASA application domain is used to illustrate our model-based approach to sensor placement.

Introduction

Sensor placement is the task of determining a set of sensors which allows the most accurate determination of the overall state of a monitored system while minimizing sensor power consumption, cost, computing power requirements, and weight. Reducing these quantities is particularly important in space-borne systems due to power and payload restrictions.

This paper describes an approach to sensor placement based upon an extension of techniques developed for sensor selection in monitoring as part of the SELMON project [Doyle et al. 91]. These techniques have two components, an information theoretic component and a model-based reasoning component. The information theoretic component captures knowledge about unusualness and informativeness of sensor data. The model-based reasoning component captures knowledge about how observed data indicates future potential system behavior. This paper focuses upon the model-based reasoning component of our sensor placement approach. In particular, we describe how model-based reasoning enables evaluation of four measures for evaluating potential sensor placements. *Sensitivity Analysis* suggests sensor placements which measure quantities which have the greatest impact upon the overall state of the system. *Cascading Alarm Analysis* suggests sensor placements which measure quantities whose changes have the potential to cause many alarms. *Potential Damage Analysis* suggests those sensor placements which measure quantities which are likely to cause permanent damage to devices in the system being monitored. *Teleological Analysis* suggests sensor placements which monitor quantities directly applicable to specified goal functionality of the system. Our approach uses a model-based simulation capability to evaluate how each sensor rates with respect to each of these measures over the behavioral space of the monitored system. These scores can then be used to generate a proposed sensor set.

This sensor placement evaluation capability provides a number of benefits. First, this evaluation capability will aid designers in the sensor placement task by facilitating evaluation of alternative sensor placements. In particular, this capability would provide a quantitative measure of tradeoffs in sensor placements which previously have been viewed only subjectively. A second benefit is that quantification of sensor placement measures will aid in design documentation by

allowing quantitative justification for sensor placements. Third, the automated evaluation capability will facilitate assessment of the impact of system design changes upon sensor placements. Finally, as a fourth benefit, this sensor placement evaluation capability can be used to aid in sensor power planning. When the utility of a sensor depends greatly upon the operating mode of the monitored device, it may be possible to reduce overall sensor power consumption by powering certain sensor suites only in limited operating modes. Because our approach measures the utility of sensors in each system operating mode, it can assist in sensor power planning.

The remainder of this paper consists of three sections. The next section begins by describing how a model of the monitored system can be used to conduct a generalized simulation and how this simulation can be used to evaluate the four sensor placement criteria. The four sensor placement criteria are then described in greater detail. The following section describes the testbed domain - the Environmental Control and Life Support System (ECLSS) for Space Station Freedom (SSF). This section contains an example illustrating how the sensor placement criteria apply to the mutifiltration subsystem of SSF life support. The final section of the paper discusses outstanding issues regarding our approach to sensor placement, compares our approach to related work, and summarizes the key aspects of our approach.

Approach

Our approach to sensor placement is shown in Figure 1 and can be described generally as follows:

1. Given nominal behavioral models of the system and a causal simulation capability, generate a behavior space for the system.

2. Apply the sensitivity, cascading alarms, potential damage, and teleological analysis for system operation over these operating modes.

3. Compute sensor placement recommendations as those with highest scores from the analyses.

Our modelling uses a representation based upon [Doyle88]. In this representation, a model consists of a number of devices; each of which has a set of associated quantities. Mechanisms represent relationships between various quantities and are instances of physical processes. Simulation proceeds by executing mechanisms whose inputs change - possibly triggering other mechanisms. Mechanisms may change quantities, potentially with an associated delay. For example, fluid moving through a pipe may change the volume of fluid at the destination, but with a delay related to the size of the pipe and the flow rate.

We now describe the Sensitivity, Cascading Alarms, Potential Damage, and Teleological analyses. The Sensitivity, Cascading Alarms, and Potential Damage analyses use the model to simulate the effect of changes in quantity values. The effects of these changes are then analyzed to produce a Sensitivity, Cascading Alarms, or Potential Damage score. The Teleological Analysis uses a specification of the functional goal(s) for the system or subsystem along with an analysis of dependencies among mechanisms in the model to produce a sensor placement score.

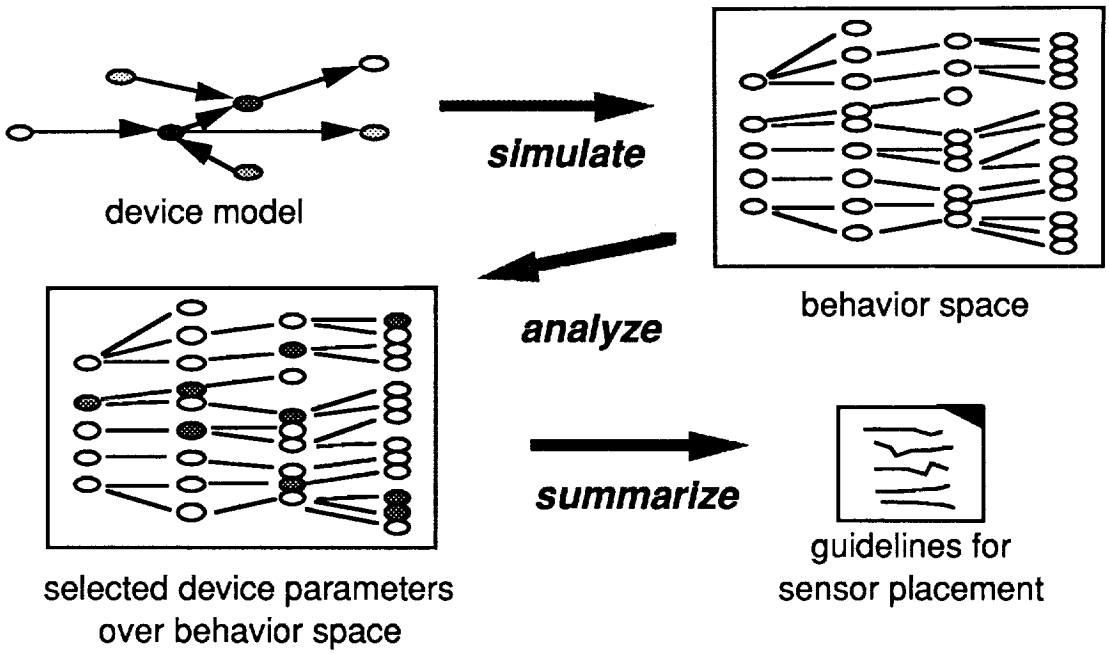


Figure 1: Model-based Simulation Approach to Sensor Placement

Sensitivity Analysis

Sensitivity Analysis measures the sensitivity of other quantities in the monitored system to changes in the current quantity. This measure depends upon information about "normal" magnitudes of change for the devices in question. For each normal operating mode of the system, the following procedure is performed. For each quantity Q $\in$ MonitorableQuantities (the set of all monitorable quantities in the model), determine nominal operating values and alarm ranges. Next compute a normalized change increase $\Delta Q+$ and decrease $\Delta Q-$ as the average amount of change between updates for that operating mode. Next, for each quantity Q, beginning with a simulation with all devices/sensors at nominal operating values, using simulation provided by the model, simulate a change ΔQ in Q, propagating this change to other quantities in AllQuantities (the set of all quantities in the model) as dictated by the model. For each such changed quantity Q' $\in$ AllQuantities, for each time the quantity changes during the simulation, collect a sensitivity score proportional to the amount of change in Q' from its normal value $Q'_{nominal}$ relative to alarm thresholds but also modified by a decreasing function of time[1]. This calculation captures the characteristic that delayed and less direct effects are more likely to be controllable and less likely to occur. Thus, a change which affected a quantity Q' but occurred slowly is considered less important. This simulation proceeds for a preset amount of simulated time. Then, for each changed quantity Q', take the maximum of the collected change score for that quantity. The sensitivity score for Q is the sum of these maximums for all the Q's. Thus, for each quantity Q, a simulated change produces a set of changescores for each other quantities in the model. The

[1]This can be viewed as an average $\partial Q'/\partial Q$ modified by a decreasing function of time elapsed and normalized for the alarm threshold for Q'.

sensitivity score for Q is the sum of the respective maximums of each of these sets[2]. The computation of the sensitivity scores is shown below:

Simulate a change $\Delta Q+$ or $\Delta Q-$ to Q beginning at time 0 and continuing to time ΔT (a user-supplied default).

For each change to a quantity Q' occurring at time T_{change}, compute a change score as follows.

let Q'_{new} be the new value for Q'

$$changescore(Q') = \frac{|Q'_{new} - Q'_{nominal}|}{|Q'_{alarm} - Q'_{nominal}|} \times \frac{(\Delta T - T_{change})}{\Delta T}$$

add this changescore to the set of collected changescores for Q'

let MaxChangeScore(Q') = the maximum of the set of collected changescores for Q'

$$\text{let } sensitivity(Q) = \sum_{Q' \in AllQuantities} MaxChangeScore(Q')$$

The overall sensitivity score for Q is then computed by summing the sensitivity scores for $\Delta Q+$ and $\Delta Q-$ weighted by relative frequency of increase vs. decrease for Q.

Cascading Alarms Analysis

Cascading alarms analysis measures the potential for change in a single quantity to cause a large number of alarm states to occur, thus causing information overload and confusion for operators. As with sensitivity analysis, cascading alarms analysis is performed for each operating mode of the monitored system. For a standardized amount of increase and decrease for each monitorable quantity Q, the effects of such a change are propagated throughout the system and the number of triggered alarms is counted. This standardized amount of change is different from the measure used in the sensitivity analysis as normal changes are not likely to produce cascading alarm patterns. The alarm count is then normalized for the total number of possible alarms. The weight of each alarm state triggered is also decreased as a function of the time delay from the initial change event to the alarm. This has the effect of focussing this measure on quickly developing cascading alarm sequences which are the most difficult to interpret and diagnose. The computation of cascading alarms scores is shown below.

Simulate a change $\Delta Q+$ or $\Delta Q-$ to Q beginning at time 0 and continuing to time ΔT (a user-supplied default); where $\Delta Q+$ and $\Delta Q-$ are functions of the distance between the nominal value for Q and the alarm value for Q in the increasing and decreasing directions respectively

---

[2]Quantities which do not change when Q is changed produce an empty set of changescores. We define the maximum of this empty set as 0 for the purpose of the sensitivity summation.

$$\text{let CascadingAlarm}(Q) = \frac{\sum_{Q' \in \text{ all quantities}} \text{InAlarm}(Q')}{\text{number of quantities } Q'}$$

where $\text{InAlarm}(Q') = (\Delta T - T_{alarm})/\Delta T$

if Q' entered an alarm range during the simulation and T_{alarm} is the earliest time Q' was in an alarm range and

$\text{InAlarm}(Q') = 0$

if Q' did not enter an alarm range during the simulation.

Potential Damage Analysis

Another measure is potential damage, which is measured in two parts - predictive potential damage and potential damage detection. Predictive potential damage measures the capability of the sensor to predict damage to devices in the system. For each device and quantity associated with that device, there is an associated operating range which is judged to be harmful to the device. Predictive potential damage analysis is performed by simulating a change in each monitorable quantity Q and scoring upon the basis of how many devices will enter harmful ranges due to the change in Q. Predictive potential damage analysis scores are moderated by the number of control points which may interdict the damage (defined as a mechanism directly influenced by a directly controllable quantity). For the causal path leading to the damaged device, for each mechanism which depends upon a control parameter, the potential damage score is reduced. The potential damage measure depends more critically upon domain-specific information beyond the schematic, as many of the potential damage scenarios involve device or subsystem interactions. The computation of potential damage scores is shown below.

Simulate a change $\Delta Q+$ or $\Delta Q-$ to Q beginning at time 0 and continuing to time ΔT (a user-supplied default).

$$\text{let PotentialDamage1}(Q) = \sum_{Q' \in \text{ all quantities}} \text{Damaged?}(Q')$$

where

$$\text{Damaged?}(Q') = \frac{(\Delta T - T_{alarm})}{\Delta T \times (\text{control} + 1)}$$

if Q' entered a damaging range during the simulation where T_{alarm} is the earliest time Q' was in a damage range and control is the number of control points in the causal chain leading to the damaging quantity value and

$\text{Damaged?}(Q') = 0$

if Q' did not enter a damage range during the simulation.

The second part of potential damage analysis is damage detection. In this measure, the model is used to simulate devices in the system entering damaging operating modes, and potential sensors are scored upon the basis of how much they change (in the same manner as the sensitivity analysis). Damage detection analysis is performed by propagating a change resulting in a device entering a damaging range, and measuring the resulting change in sensor reading as in the sensitivity analysis. Those sensors which change more significantly to indicate the damaging device state are scored higher by the damage detection analysis.

Let $\Delta Q'+$ or $\Delta Q'-$ be changes sufficient to cause Q' to enter a device damaging range.

Simulate a change $\Delta Q'+$ or $\Delta Q'-$ to Q' beginning at time 0 and continuing to time ΔT (a user-supplied default).

$$\text{let PotentialDamage2(Q)} = \sum_{Q' \in \text{ all quantities}} \text{Changescore(Q)}$$

The final measure is teleological analysis, which does not use the model-based simulation capability. Instead, the teleological analysis examines the mechanism dependencies to produce a sensor placement score. In some cases, some aspects of the purpose of the monitored system can be specified in terms of quantities in the model, some of which may be monitorable quantities. The teleological analysis measure suggests measurements of quantities most closely linked to the functional goals of the system being monitored. In this measure, those quantities directly mentioned in the functional specification of the system are scored highest, those quantities directly influencing these quantities are scored next highest, etc. The exact computation of the teleological measure involves backtracing the causal graph. Directly monitorable quantities appearing in the goal description receive a score of 1. For each mechanism affecting the goal quantity, a teleology score inversely proportional to the number of such mechanisms is then divided equally among the inputs to the mechanism. Thus, if there are M mechanisms affecting a goal quantity, and one of these mechanisms has N inputs, each such input will receive a score 1/MN. Note that multiple causal influence paths will combine additively. While this process proceeds recursively for the mechanisms potentially influencing the inputs to this mechanism, each level is multiplied by 1/D where D is the number of mechanisms distant from the goal quantity.

The sensor placement scores computed by evaluating these four measures can be used in two ways. First, design engineers can use these scores directly to aid in their decision making process. Second, these scores can be combined in a utility function to rate the desirability of each potential sensor placement. The sensor placement task can then be viewed as maximizing the utility function within certain cost constraints (e.g. weight, power consumption, computing resources, cost, etc.). While we are currently investigating the sensor placement task within the context of the first approach (e.g. development of a sensor placement evaluation tool), we expect the resulting evaluations to be applicable to the second task.

Domain and Status

Our sensor placement approach is being tested upon the water reclamation subsystem of the Environmental Control and Life Support System (ECLSS) for Space Station Freedom. A model describing the behavior of the multifiltration (MF) subsystem in

terms of fluid flow and heat transfer has been constructed. This model was developed via a combination of study of design documentation (i.e. schematics, etc.) and consultation with domain experts (e.g. the operators of the testbed). This model has been validated by comparison against actual data from the subsystem testbed undergoing evaluation at the Marshall Space Flight Center in Huntsville, Alabama. We are also in the process of extending our model to cover more of the ECLSS subsystems.

Figure 2 below shows the ECLSS multifiltration (MF) subsystem. The MF subsystem consists of two main parts - the sterilization loop and the unibed assembly. In the MF subsystem, the water first passes through a pump at the inlet to the system. Next, the water passes through a coarse filter before entering the sterilization loop. In the sterilization loop the water is heated in the regenerative heat exchanger and then by the in-line heater after point 3. The in-line heater has only a coarse temperature control and thus the water temperature at point 4 may differ as much as 10° F from the goal of 250° F. Within the sterilizer reservoir, the temperature of the water is maintained more accurately at 250°F for about 9 minutes. In the second portion of the subsystem, the water passes through a set of unibed filters designed to remove particulate contaminants from the water. Possible sensor types are flow rate, water pressure, temperature. Possible sensor locations are indicated in Figure 2 by numbered ovals.

Specified functional goals are:

1. maintain processed water at 250°F in sterilizer reservoir for 9 minutes; and
2. maintain water flow through the unibed of at least 15 mL/minute.

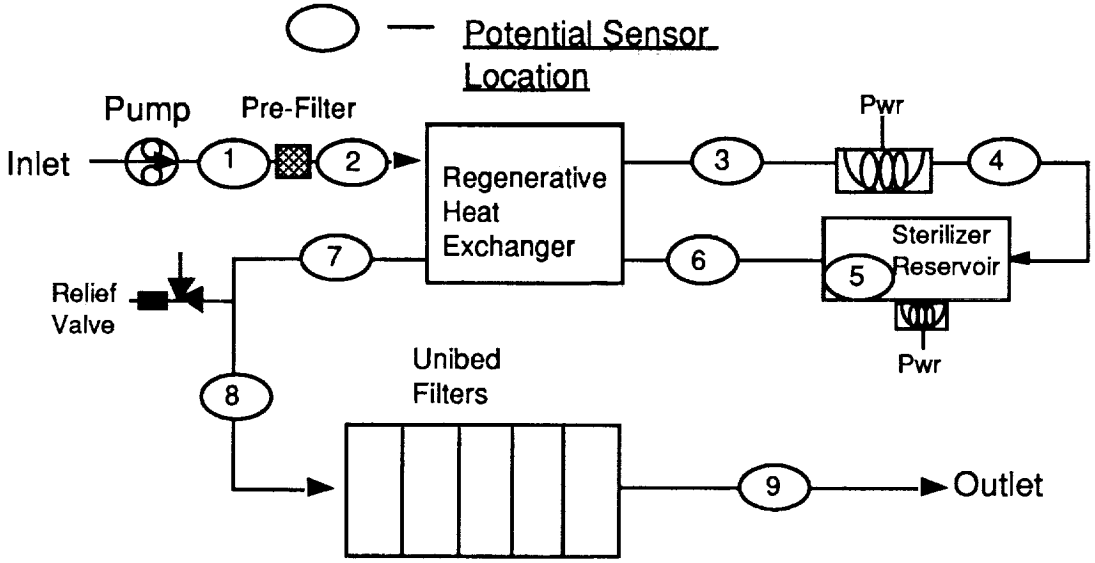


Figure 2: Multifiltration Subsystem

In this example, Damage Detection Analysis suggests placing a temperature sensor at point 4. This is because if the in-line heater overheats and enters a damaging temperature range, it would cause the water flowing through the in-line heater to be heated to a higher temperature than normal, thus causing a significant temperature increase at point 4.

Sensitivity Analysis scores highly the placement of a flow rate sensor anywhere in the MF because the flow rate affects many other quantities. The flow rate affects the temperature at each of the potential sensor placement points in that the flow rate affects the heat loss in the pipes. The flow rate also affects the speed of the temperature propagation from the fluid flow (e.g. delays for temperature changes).

Sensitivity Analysis also suggests the more specific placement of a pressure sensor near the relief valve at point 7. This is because the relief valve is pressure controlled; if the pressure at point 7 is above 40 psig, the relief valve will open and drastically change the system behavior. The opening of the relief valve would cause an immediate significant pressure loss, as well as significantly affecting flow in the MF subsystem.

Teleological Analysis suggests placing flow rate sensors to verify the flow of water through the unibeds as the flow rate is directly mentioned in the goal specification. Teleological Analysis also highly scores a flow rate sensor in the sterilizer reservoir, as this quantity determines the time spent by the water in the sterilizer reservoir. Teleological Analysis also scores slightly lower, flow rate sensors at any of the other locations, as in normal operation the flow rate is the same at all of the potential sensor locations. Finally, Teleological Analysis also suggests placement of a temperature sensor for the sterilizer reservoir, as this quantity appears in the functional goal specification of the system.

While the MF subsystem model has been completed and verified against actual testbed data, the sensor placement scoring algorithms are currently being implemented. When finished, the sensitivity, cascading alarms, potential damage, and teleological measures will then be tested on the current testbed configurations with domain experts evaluating the accuracy and utility of the sensor preference criteria.

Discussion and Conclusion

The research described in this paper is preliminary, hence there are a number of outstanding issues left to future work. One issue is that of the computational expense of simulation. While we are currently implementing a generalized simulation capability for all behavioral modes, this approach may be intractable for more complex systems Hierarchical simulation schemes and/or Monte Carlo sampling of the behavior space are currently being examined as approaches to dealing with this problem.

Another difficulty is determining how far forward to simulate for the sensitivity, cascading alarms, and potential damage analyses. Additionally, how large of a quantity change to propagate is another issue. Currently, both of these are parameters which may be changed by the user. However, ideally, the system would be able to determine appropriate values for these parameters.

Our work in sensor placement for monitorability is also relevant to issues in design for diagnosability. Because methods for ensuring diagnosability depend upon completeness of fault models, if fault models are incomplete, sensor placement based solely upon diagnosability criteria may make it difficult to detect and diagnose unforseen misbehaviors. Thus an approach to sensor placement based only on criteria of diagnosability may result in sensor configurations which do not support safe system operation. This is an important point about the difference between

monitoring and diagnosis, or the difference between anomaly detection and troubleshooting. The goal of sensor placement for diagnosability is to ensure that sensor data to support the diagnosis of known faults will be available to operators. However, as the history of the Voyager mission tells [Laeser et al. 86], the potential for unforeseen faults with obscure manifestations always exists. There will be no substitute for years of experience and troubleshooting expertise on the part of domain experts in handling successfully these most difficult and potentially fatal cases.

In accordance with this observation, we are developing a comprehensive approach to sensor placement based upon both diagnosability and monitorability. This paper has described our work in design for monitorability in which the goal is to provide operators with high information content sensor data and/or sensor data which report on critical causal pathways in the device, without reference to fault models. Information-theoretic monitorability criteria capture knowledge about unusualness and informativeness of sensor data and the four model-based reasoning monitorability criteria described in this paper capture knowledge about how observed data indicates future potential system behavior. Although sensor placement based on monitorability does not provide a comprehensive solution to the problem of unforeseen faults, the intent is to provide experienced operators with the most generally informative sensor data and to avoid having sensor placements be wired into an inevitably incomplete set of fault models.

In a parallel effort, we are also examining the use of fault models to evaluate sensor placements from a diagnosability standpoint [Chien & Doyle91]. These methods complement the sensor placement for monitorability approach described in this paper. Thus, in those cases where fault models exist, a diagnosability analysis will incorporate such knowledge in sensor placement recommendations. For example, the types of interactions addressed by the Cascading Alarms and Potential Damage Analyses can be focussed by fault model information (by indicating which such interactions are likely to occur). Thus, fault model information is used in the diagnosability analysis. However, because monitorability analysis does not use fault models, it offers better coverage of unforseen faults.

Other work on sensor placement includes [Scarl91a, Scarl91b]. Scarl's work focuses upon two issues: 1) discrimination of faulty sensors (and hence faulty sensor data) from the occurrence of regular system faults; and 2) deriving minimal sensor sets to cover known fault sets. In contrast, our work focuses upon quantifying how well proposed sensors meet general monitorability criteria. Other relate work includes work on teleology and model-based reasoning [Sticklen et al. 89, Franke89, Sun & Sticklen90]. Sensor placement for monitorability is also related to design for testability [Wu88, Shirley88]. In this work, design constraints for digital circuits are derived from testability constraints. This work differs from our work in several respects. First, we are concerned with monitoring systems with continuous quantities. Second, issues of time criticality arise in our domain. And third, we are also concerned with monitoring in non-faulted modes.

This paper has described a model-based reasoning approach to evaluating sensor placements from the standpoint of monitorability. In this approach, potential sensor placements are evaluated using four criteria. Sensitivity Analysis suggests monitoring of quantities which, if changed, significantly affect overall system state. Cascading Alarms suggests monitoring of quantities which may lead to rapidly developing alarm sequences. Potential Damage suggests sensors which monitor quantities which predict or detect damage to devices in the monitored system. And

Teleological Analysis suggests monitoring of quantities more directly causally linked to the specified goal functionality of the system.

Acknowledgements

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. We would like to thank Jay Wyatt of Marshall Space Flight Center for innumerable discussions regarding the operation of the hygiene subsystem.

References

[Chien & Doyle91] S. A. Chien and R. J. Doyle, "A Model-based Reasoning Approach to Sensor Placement for Diagnosability," Internal Working Paper, Artificial Intelligence Group, Advanced Information Systems Section, Jet Propulsion Laboratory, California Institute of Technology, 1991.

[Doyle88] R.J. Doyle, "Hypothesizing Device Mechanisms: Opening Up the Black Box," Technical Report 1047, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, June 1988.

[Doyle et al. 91] R. J. Doyle, U. M. Fayyad, D. Berleant, L. K. Charest, L. S. Homem de Mello, H.J. Porta, and M.D. Wiesmeyer," Sensor Selection in Complex Systems Monitoring Using Information Quantification and Causal Reasoning" in Recent Advances in Qualitative Physics, B. Faltings and P. Struss (eds.), MIT Press, 1991.

[Franke89] D. Franke, "Representing and Acquiring Teleological Descriptions," Proceedings of the 1989 Workshop on Model Based Reasoning, August 1989, pp. 62-67.

[Laeser et al. 86] R. P. Laeser, W. I. McLaughlin, and D. M. Wolff, "Engineering Voyager 2's Encounter with Uranus", *Scientific American*, 255(5):36-45, November 1986.

[Scarl91a] E. Scarl, "Diagnosability and Sensor Reduction," Proceedings of the Workshop on AI, Simulation, and Planning in High Autonomy Systems, Cocoa Beach, FL, 1991.

[Scarl91b] E. Scarl, "Monitoring and Diagnosis: Stress from Weakened Environmental Knowledge," Proceedings of the 1991 IEEE Conference on Robotics and Automation, Sacramento, CA, 1991.

[Sticklen et al. 89] J. Sticklen, B Chandrasekaran, and W. Bond, "Applying a Functional Reasoning Approach for Model-based Reasoning," Proceedings of the 1989 Workshop on Model Based Reasoning, August 1989, pp. 165-176.

[Sun & Sticklen90] J. Sun and J. Sticklen, "Steps Toward Tractable Envisionment via a Functional Approach," Proceedings of the 1990 Workshop on Model-based Reasoning, August 1990, pp. 50-55.

[Shirley88] M. Shirley, "Generating Circuit Tests by Exploiting Designed Behavior," Technical Report 1099, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, December 1988.

[Wu88] P. Wu, "Test Generation Guided Design for Testability," Technical Report 1051, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1988.

Distributed Environmental Control

Gary A. Cleveland
McDonnell Douglas Space Systems Company - Space Station Division
5301 Bolsa Avenue, Huntingon Beach, CA 92647
(714) 896-3311 x7-0311
cleveland%ssdvx1.decnet@mdcgwy.mdc.com

1.0 INTRODUCTION

We present an architecture of distributed, independent control agents designed to work with the Computer Aided System Engineering and Analysis (CASE/A) simulation tool. CASE/A simulates behavior of Environmental Control and Life Support Systems (ECLSS). We describe a lattice of agents capable of distributed sensing and overcoming certain sensor and effector failures. We address how the architecture can achieve the coordinating functions of a hierarchical command structure while maintaining the robustness and flexibility of independent agents. These agents work between the time steps of the CASE/A simulation tool to arrive at command decisions based on the state variables maintained by CASE/A. Control is evaluated according to both effectiveness (e.g., how well temperature was maintained) and resource utilization (the amount of power and materials used).

1.1 Motivations and Criteria

We employ five criteria in designing and building this control system.

1) The controller is to work with the CASE/A simulation system.

2) The architecture should introduce as little new vocabulary as possible to describe the systems being simulated by CASE/A and their control. We wish to keep the usability of the CASE/A system high, especially among its current user community.

3) The controller must coordinate diverse and conflicting functions among sensors and effectors that are spatially remote, work in a system with significant time delays, and are subject to certain types of faults.

4) Control of the system must degrade gracefully in the face of several types of faults.

5) The control architecture should be modular so that new controllers can be constructed to match new simulations without changing any large part of the basic scheme.

6) The control mechanism should make use of a parallel processing model of computation. We do not want to eliminate the option of a parallel implementation.

1.2 General Description of this Work

It is fairly straightforward to control a given device at about the level at which one would describe a thermostat or proportional control. One uses rules such as, "When the temperature gets above a certain point, turn on the air conditioner," and, "As the vibration increases, decrease the rate of spin by an amount depending on the severity of the vibration." These examples and our viewpoint are meant to be consistent with the control found in the field of robotics; sensors provide input, effectors receive output, controllers map the former to the latter.

This effort takes a more generalized view of sensors and effectors, introducing explicit levels of control and allowing control to be described uniformly at all levels. Communication between the control levels is carried out using the same set of constructs as are the sensing and effecting at the lowest level. In effect, a higher level controller "senses" the information that the lower level controllers make available. The introduction of multiple levels of control is the key factor in being able to address control issues that

span several sensors, and in being able to produce coordinated control over a number of effectors. This abstraction also provides an ability to step away from the hardware a bit and deal with the control problem in an intuitive manner. For example, suppose a pair of sensors measure air pressure and the partial pressure of oxygen, respectively. If our control problem is built around the percent of oxygen in the air, then the control algorithm will be made less clear by the extra computation. We are proposing that this extra translation be extracted from the basic control algorithm. Similarly, levels of control allow sensors such as "AirLock Nominal" to be constructed. Such a "virtual" sensor might look at a half-dozen other sensors (both virtual and hard sensors) using a complex algorithm before actually determining that the airlock status is nominal. In both cases, the translational and analytical work is separated from the control algorithm that uses the analysis.

1.3 Relationships to Other Work

The official description of the CASE/A simulation system is found in the CASE/A User's Manual [CASE/Aa] with additional insights and details provided by the Programmer's Manual [CASE/Ab]. Of special value in understanding our work is the detailed description of how CASE/A handles time steps.

More to the point of our effort, the reader should consult the work on subsumption architectures at MIT ([Brooks], [Connell]) from which we have adopted much of our communication model and protocol. This communication scheme has been modified to address some of the same issues as Henderson's work ([Henderson84], [Henderson90]). We perceive Henderson's work as an attempt to rise above limitations in the MIT work coming from the low level at which those systems are built. We have adopted Henderson's scheme as far as our simple sensors and effectors made desirable. We contrast the complexity of Henderson's vision sensors to that of our thermometers and air pressure gauges.

One may note the similarity in some aspects to neural networks [Hopfield]. Three comments pertain. First, our graph of agents does employ a communication protocol similar to that used between neurons. Second, the model of computation carried out by our agents leans more towards the symbold than that found in neurons (making use of stored state variables, for example.) Third, it is not our intention to endow our agents with any learning potential.

Mention should also be made of experiments in the area of reactive intelligent control such as performed by Agre and Chapman [Agre]. By using the building blocks described below, our long term goal is to be able to construct reactive controllers which use knowledge at a level similar to that found in Agre and Chapman's Pengi system.

1.4 Organization of the paper.

Section 2.0 presents a low level view of communication with the CASE/A simulation tool. Section 3.0 describes the lattice of controlling agents and their functions. Section 4.0 describes the evaluations to be performed on the simulation runs controlled using this architecture.

2.0 COMMUNICATING WITH THE CASE/A SIMULATION SOFTWARE

2.1 The CASE/A Simulation Software.

The Computer Aided System Engineering and Analysis (CASE/A) modeling package is an Environmental Control and Life Support System (ECLSS) and Active Thermal Control System (ATCS) analysis and trade study tool. The package is written in FORTRAN and supports the construction and analysis of ECLSS and ATCS models by offering primitive units such as pumps, heat exchangers, etc., that can be linked together to form the desired models. The primitive units are referred to as *components* and the links between them, for the purposes of this presentation, are called *streams*. Streams themselves are discussed in terms of the *constituents* that flow through them. Oxygen, water, and other materials are examples of constituents. The *properties* of streams are also of interest. The most often discussed properties are temperature and pressure.

Each component in a CASE/A model has a function that computes, at every time step, the values for the output streams given the values found at the input streams. The CASE/A package visits the various components in a model, finding new values for the streams. Naturally, the output

streams of some components are the input streams for others. Since there are cycles in the models, CASE/A will visit some streams and components repeatedly. While so doing, CASE/A is attempting to find a solution that satisfies all of the interconnected components. Some components may be visited numerous times on a given time step before convergence is achieved.

CASE/A also supports its own version of controllers that can be linked among the streams and components and which support a language similar to a zero-register (stack-based) assembly language for forming and carrying out control decisions. Unfortunately, these controllers can only sense one value at a time and can only affect one value at a time. This eliminates the possibility of any straightforward scheme for coordination among sensors or effectors. These controllers do, however, provide the basics of the model of communication between CASE/A and the controller we are building.

2.2 Communicating with CASE/A

As was hinted in the previous section, the streams (constituents) in CASE/A provide a natural correspondence to the sensors that we desire to create. The components provide our vocabulary of effectors as well as providing more sensors of interest (e.g., pump flow rate). As effectors, we may set a pump flow rate based on the temperature of a water line. It remains only to create an import/export mechanism enabling the values to be moved between the existing CASE/A package and the newly created controller. It turns out that for the VAX/VMS system where CASE/A resides, communication between FORTRAN and the controller's home language of Ada is straightforward.

The timing of the communication is also straightforward but still bears discussion. To control a physical system, we would be faced with accepting and reacting to asynchronous sensor signals. To the degree possible, we wish to work with this model even though the CASE/A system works in discrete time steps. CASE/A will transfer program control to the controller only between those time steps. The resulting communication scheme has CASE/A passing a number of sensor signals to the controller all at the same time but in no particular order. Because the controller processes these signals using an asynchronous model, it sometimes happens that control (effector) signals are generated and then overridden before reaching a final form. Overrides, as discussed below, are one form of communication between control agents. After all the control signals have been generated and have stabilized, the packet of signals is sent back to CASE/A for another time-step iteration.

3.0 THE AGENT ARCHITECTURE

3.1 Control Agent Description

The primary unit of control in our architecture is the *agent* as depicted in Figure 1. Each agent has ports to accept sensor input and to produce effector commands. Command mappings from the sensors to the effectors are produced by one of a number of algorithms available to the agent. Each such algorithm may be viewed as one part of a production (rule-based) system. These control algorithms are local to the agent and operate independently from those of other agents. Each agent can also maintain memory of past sensor values and effector commands so that trends may be noticed and complex actions requiring a schedule of sub-actions may be effected. The set of sensor ports and the set of effector ports are redundant in that there may be several means of deriving the same information or issuing the same command from different subsets of the available ports. Coordinated behavior is achieved spatially by using connections between the ports as communication lines and temporally by utilizing the internal memory of the agents.

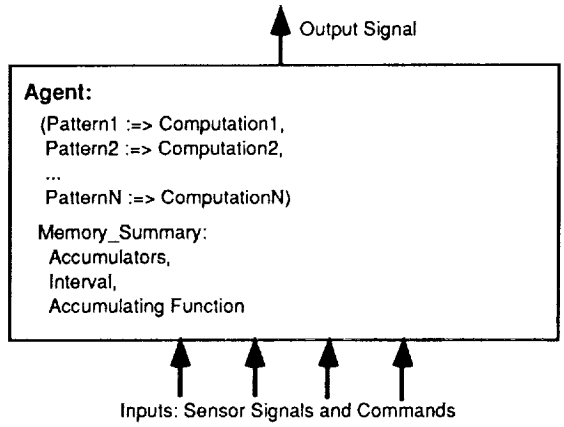


Figure 1: Structure of Agents (with Memory)

3.2 Coordination of Agents

All agents are arranged in a tangled hierarchy (directed acyclic graph) with the sensor and control signals traveling up through the graph. Topologically, this is identical to Brooks' networks of subsuming agents and still similar to Henderson's hierarchy of logical sensors. All the control agents below a given agent are treated as sensors while all the agents above a given agent are treated as effectors. The leaves of the graph represent the actual sensors (as found in the CASE/A simulation) while the roots of the graph represent the actual effectors. Functionally, this arrangement is also similar to Henderson's work because the higher level agents have control (through overrides and subsumption) over those at lower levels. The difference lies in the arrangement of connections in the graph.

The network of agents is static during a simulation run. Dynamic behavior is obtained from this static net when a given agent chooses to invoke a different algorithm for sensor analysis and effector control signal generation. This parallels Henderson's work with the important distinction that sensor and command signals are combined in one communication channel.

Generally, the leaves of the graph directly relate to sensors whose signals are transmitted through the interface from the CASE/A simulation. Control is accomplished by manipulating these sensor signals as they pass up through the graph. Signal manipulation takes the form of computing new signals to pass on from those received. Some of the signals received will correspond closely to physical values produced by the simulation while others will be better interpreted as control or context signals. The two types of signals are treated alike.

While these "control signals" are treated the same as the "sensor signals," one may view their treatment from several viewpoints. The first is that the controlling agents reside in the agent network "above" those agents that they control. The agents higher in the graph produce control signals that override the signals generated by the lower agents. This view corresponds to subsumption as put forth by Brooks *et al.* The other view is that the controlling agents reside "below" the controlled agents. By providing different inputs, the lower level agents can influence the behavior of the upper level agents. As with Brooks, we prefer that the higher level agents might have knowledge of the graph below them but not that any lower level agent should ever have knowledge of the graph above.

Depending on current and previous sensor (and control) values, various computations may be used to create the signals that will continue up the graph. The choice of algorithm may also depend upon estimates of confidence in the signals being passed in. Actual algorithm selection is performed by a simple pattern match against current input and stored values.

3.2.1 Sensing

Agents sense declared numeric values within the CASE/A simulation. All sensors are defined in terms of the structures (usually constituents of streams) that CASE/A already maintains although communication among agents depends on slightly different streams. Using this scheme, three more complex sensing behaviors can be constructed.

3.2.1.1 Grouping

Homogeneous groups of identical sensors in parallel can be used as the simplest means of obtaining fault tolerance. Most of the time a group of such sensors is viewed as a single sensor, producing a single reading derived from the combination (usually the average) of the readings of the individual sensors. A complex sensor of this type needs to have some facility for dealing with failed individuals. The reading from the combination of sensors can usually be assigned a higher confidence value than that of any of the individuals.

3.2.1.2 Fusion or Virtual Sensing

Heterogeneous groups of sensors may also be constructed and represented as a single, combined sensor as shown in Figure 2. This type of complex sensor can produce a "sensed" value derived from but not directly related to any physical measurement. Most of the "control" agents take this form.

3.2.1.3 Integration, Trends, and Time Averages

Sensor agents with memory can store past readings in order to produce values for totals, trends, and averages over time. The outputs

from these sensors are referred to as control signals for the sake of uniformity. In fact, these signals are usually piped straight to the input of another agent that treats the signal as another type of sensor.

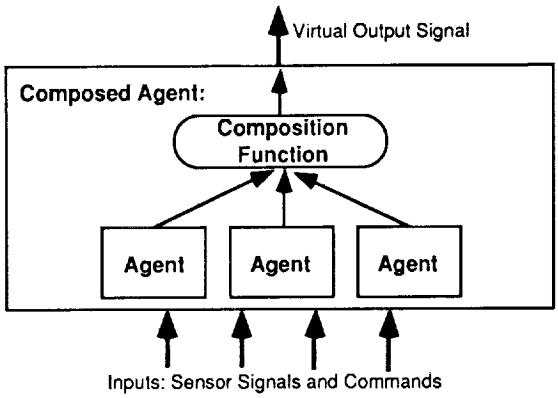


Figure 2: Agent Composition for Coordinated Control and Virtual Sensing

3.2.2 Effecting

Carrying out the control decisions is carried out by communicating those decisions to CASE/A, in the form of numeric values corresponding to desired settings for component attributes (e.g., flow rate of a pump). Most of the important pieces needed for coordinated control have already been introduced with the discussion of the controllers above. It remains to show how those controllers can be used to carry out complicated behaviors.

3.2.2.1 Coordinated actions

3.2.2.1.1 Among Agents

Coordinating actions among agents is straightforward given the arrangement of controllers in the graph as described above. A controlling agent merely outputs a value that signals a certain context has been entered. The agents being coordinated must have this context preprogrammed as one of the patterns to which they respond. We are again relying upon experience to show whether the number of such patterns is prohibitive and whether coordinated actions will need to show more flexibility than can be achieved with the scheme just outlined.

An alternative means of rendering such coordination would be to have agents watch each others' command streams and act accordingly. This second scheme eliminates the need for the (extra) coordinating agent but even more strongly begs the question of pattern complexity. Probably, the domain and the specific behavior being programmed will determine which method is used.

3.2.2.1.2 Through Time

For coordinating actions through time we rely upon the sensor agents' capability for collecting accumulated data such as averages and trends. Such sensors can "count time" as well as watch the behavior of sensed values. Agents already have the ability to respond to changes in the sensed environment and can thus respond to other agents' actions (e.g., after Agent1 starts the motor, Agent2 should open the valve). The agents with the time memory will allow scripted behaviors among one or more agents. A scripted behaviors resides within a single agent and is carried out when that agent sends appropriate signals (just as all others) to other agents. For example, Agent1 and Agent2 should both turn on motors for five minutes. The beginning and end of the five minutes is called out by Agent3, possibly by claiming to sense an imbalance in a hydraulic line that goes away after five minutes.

3.2.2.2 Virtual Effectors

Agents that are somewhat removed from the actual control of the components accessible from CASE/A may be referred to as "virtual effectors." An example is a controller that is programmed to "raise the temperature" but does not have direct access to the CASE/A program. Such a controller should be in a position of communicating to some set of heaters, fans, lights, etc that can be coordinated according to the current set of tradeoffs. In fact, this is a complementary view of the coordination of sensors mentioned above. In one, the system tells the sensors what to be sensitive to and in the other, we view this from the coordinating agent's vantage of working towards some purpose.

3.3 Fault Tolerance

Fault tolerance is becoming one of the most important issues in controlling complex systems. For large systems, it is not practical to try to eliminate all faults. For this reason, nearly all fault tolerance arises from the introduction of redundancy into the system. The basic scheme is to have one part of the system stand in for another part when it fails. This leaves us with the two questions concerning what parts may fail such that the

system still functions reasonably and how is it that this "reasonable" or "acceptable" behavior is achieved.

For the current version of the control system, we address faults in the CASE/A sensors and effectors only. When a sensor fails, it stops broadcasting signals of any kind. When an effector fails, it does so by acting as though it were receiving random commands or by "sticking" to either a maximum or minimum command value. We assume the control system components themselves to be above suspicion. As a further simplification in the first implementation, we assume that there exists some diagnostic program for the sensors that can tell us which sensor is failing. This allows us to concentrate on the actions the controller should take given a (single) failure rather than revisiting the subject of automated diagnosis. With this amount of information, we are also capable of quickly inferring faults in the effectors when they occur. Again, our emphasis is on the treatment of these faults more than the location. It may be that in some cases we can not specifically locate a fault but can take steps to work around it.

3.3.1 Sources of Redundancy

Two of the forms of redundancy that the system uses have already been mentioned. The first is the parallel replication of identical sensors. Assuming that only one sensor fails at a time, we can always achieve a reliable reading. We still lose information, of course, with each loss of a parallel sensor. The second form of redundancy is buried within the coordinating controllers. The coordinating agent must shift the control commands from a failed effector to those still working. For example, if a heater fails, one might turn a fan down in order to conserve more of the available heat.

Other forms of redundancy are also exploitable. Given that the controllers are able to carry out sequences of actions through time, one may rely upon the inertia of the system to achieve what an effector normally would. When a water pump fails, one may use the water stored in a holding tank for a fixed amount of time.

3.3.2 Sensors

Because of the way sensor signals are processed through a pattern match and then computation, it is possible to invoke different processing algorithms based on the availability of a sensor signal as well as based on the signal's value. When a sensor fails, those controllers that rely on the signal switch to contingency algorithms or get overridden by controllers designed to watch for just such an occurrence. In many ways, the lack of signal from a sensor is treated exactly as if the signal values were out of some range; new action is triggered.

The one truly unusual way that sensors may be used in this whole architecture is to make use of the "predictive" properties of those controllers that have memory for charting averages or trends. With the assumption that an average or trend will continue, a controller may issue commands for some time based on predicting what the important signals ought to be. This behavior is an example of using the system inertia for sensing purposes. In fact, this behavior may be invoked even without loss of a sensor.

3.3.3 Effectors

The one comment that remains to be made with respect to the effector is in addressing diagnosis. It will sometimes be the case that a sensor will begin to report unbelievable values while still checking out as operative. In this case, the system must identify the effector that is malfunctioning. This will be a non-trivial chore as several effectors are sure to affect any given sensor. Possibly by modulating the control values sent to the effectors, the site of the malfunction can be deduced. Another promising technique is that of set covering. Even if we can only pare down the list of possible failures, that may be enough to allow the controllers to use the operational effectors to offset to ill effects of the failed effector.

3.4 An Example

The example control network that we return to repeatedly is that of a thermostat. Despite its simplicity, the thermostat can be used to demonstrate most of the interactions we face.

Supposing we have a thermostat connected to a thermometer for a sensor (Thermostat_Sensor1) and a heater for an effector as shown in Figure 3.

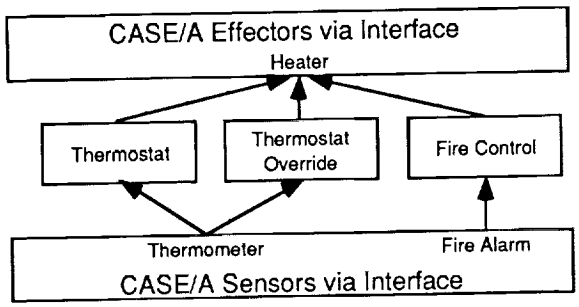


Figure 3: Successive Overrides of Control Signal

The behavior of the thermometer is to hold the temperature at approximately 25°C. The basic controller for the thermostat works under a control law expressed in patterns and commands as shown in Table I. Granted, this control law may result in an undesirable oscillation from one output command to the other.

Pattern:		Output
Thermostat_Sensor1 < 25	⇒	ON(50%)
Thermostat_Sensor1 >= 25	⇒	ON(0%)

Table I. Simple Control Law

Assuming that 50% of the heater capacity is sometimes insufficient to keep the area warm, we see that our design is not finished. We decide to add a second control agent which provides no output signal until the area we are warming becomes thoroughly cold. Under these conditions, the second agent overrides the signal of the first. The second agent has a control law similar to the first as is shown in Table II.

Pattern:		Output
Override_Sensor1 < 5	⇒	ON(100%)
Override_Sensor1 >= 5	⇒	Nil

Table II. Override Control Law

Obviously, this capability could have been included in the control law of the first agent. That it was not is due to our notions of modularity. When circumstances change significantly (e.g., water in the room is about to freeze), it is appropriate that another controller watch for the change and take appropriate action. An example of a more significant change would be the presence of a fire in the area. An additional agent is introduced to monitor the fire alarm. Alarm_Sensor1 referenced in the control law shown refers to the Fire Control agent's first sensor. When a fire is spotted, the heater is turned off as any electrical appliance should be. This law is shown in Table III.

Pattern:		Output
Alarm_Sensor1 = OFF	⇒	Nil
Alarm_Sensor1 = ON	⇒	On(0%)

Table III. Fire Alarm Control Law

We note that the actual implementation of the override function is carried out by the Heater agent which communicates through the interface to issue commands directly to CASE/A. The Heater agent contains a control law that prefers the override signals (if present) to the basic command signal. This agent's control law is shown in Table IV.

Pattern:		Output
Heater_Sensor3	⇒	Heater_Sensor3
Heater_Sensor2	⇒	Heater_Sensor2
Heater_Sensor1	⇒	Heater_Sensor1

Table IV. The presence and absence of signals determines behavior.

As a last note, we comment that this entire complex controller for the Heater can be encapsulated by another agent with two sensor inputs and one effector output. Although we may not have landed on the exact control laws that we need, we have built the structure needed to support the behavior necessary. We have also built this structure incrementally and in modular form that can be repeated and further built upon. If modifications are necessary, they should be also well-contained.

4.0 EVALUATION OF THE ARCHITECTURE

4.1 Effectiveness of Control

Part of any experiment must be an evaluation of the results of the experiment. In our case, this may be roughly voiced as, "Have we provided a sufficiently robust and realistic control that the users of the CASE/A system have benefited significantly?" We may also ask if the results of this experiment might apply to other domains as well. It has been our design goal to ensure that they do. Since all of the values and control parameters are numeric, the effectiveness may be measured using standard statistical measure. Due to the unavailability of these measures for other control systems and architectures, comparisons may not be possible.

4.1.1 Setpoint Accuracy

The primary criterion of our control system behavior must be, "Does it do what we told it to?" Given a list of setpoints for the values in the system, are we able to control the system so that those values are at or near those setpoints? Does this remain true over time?

4.1.2 Prediction and Anticipation

Is there a means of informing the system of a major change of context or of operating mode such as harvesting a crop in a plant chamber? How well does the system cope with this? How large a disturbance can it handle?

4.2 Resource Utilization

Given that the system is behaving properly or at least acceptably (i.e., within some limits), we wish to observe the cost of the behavior in terms of resource consumption.

4.2.1 Resource Usage and Local Optimization

Once an acceptable behavior has been achieved it should be possible to make small changes to the system's behavior and measure the change in terms of resource utilization. Typically, resources will include power, crew time, and materials (oxygen, water, etc). Both types of resource trade-offs can be addressed using statistical decision making tools.

4.2.2 Resource Trade-Offs

Some importance or relative cost for the various resources must be assigned since there will be a number of control strategies meeting the system requirements but using different amounts of the various resources. In this case, time must also be considered a resource as some tasks may be carried out with less material commitment if done more slowly. Within the limits of parameter perturbation, the control system can be used to investigate resource trade-offs. We are not attempting to automate the more involved notion of trying out entirely different control strategies.

If the behavior of the system is specified within relatively large intervals, it should also be possible to trade the system effectiveness for resource conservation. For example, if a temperature can be held on the low end of its acceptable range, we may be able to avoid using a heater. This might be accomplished by pumping waste heat from the living quarters and saving electrical power both in avoiding use of the heater and in avoiding excessive use of coolers elsewhere in the environment.

BIBLIOGRAPHY

[Agre] Agre, Phillip E. and David Chapman, "Pengi: An Implementation of a Theory of Activity", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Morgan Kaufman Publishers, 1987.

[Brooks] Brooks, Rodney, "A Robust Layered Control System for a Mobile Robot, *Journal of Robotics and Automation*, March 1986.

[CASE/Aa] Integration Analysis CASE/A User's Manual, Sept. 1989, McDonnell Douglas Report MDC W5074-4.

[CASE/Ab] Integration Analysis CASE/A Programmer's Manual, Aug. 1990, McDonnell Douglas Report MDC W5146-3.

[Connell] Connell, Jonathon, "A Colony Architecture for an Artificial Creature," MIT Tech. Report AI-TR 1151, 1990.

[Henderson84] Henderson, Thomas and Esther Shilcraft, "Logical Sensor Systems," in *Journal of Robotic Systems* **1**(2), pp 169-193 (1984), John Wiley & Sons, Inc.

[Henderson90] Henderson, Thomas and Rod Grupen, "Logical Behaviors," in *Journal of Robotic Systems* 7(3), 1990, John Wiley & Sons, Inc.

[Hopfield] Hopfield, J.J. and D.W. Tank, "Computing with Neural Circuits: A Model," *Science* Vol. 233, Aug. 1986.

N93-11925

A STATE-BASED APPROACH TO TREND RECOGNITION AND FAILURE PREDICTION FOR THE SPACE STATION FREEDOM

Kyle S. Nelson
Research Scientist
Honeywell Systems and Research Center
3660 Technology Dr.
Minneapolis, MN 55418

George D. Hadden, Ph.D.
Research Fellow
Honeywell Systems and Research Center
3660 Technology Dr.
Minneapolis, MN 55418

1.0 ABSTRACT

A state-based reasoning approach to trend recognition and failure prediction for the Attitude Determination, and Control System (ADCS) of the Space Station Freedom (SSF) is described. The problem domain is characterized by features (e.g. trends and impending failures) that develop over a variety of time spans, anywhere from several minutes to several years. Our state-based reasoning approach, coupled with intelligent data screening, allows features to be tracked as they develop in a time-independent manner. That is, each state machine has the ability to encode a time frame for the feature it detects. As features are detected, they are recorded and can be used as input to other state machines, creating a hierarchical feature recognition scheme. Furthermore, each machine can operate independently of the others, allowing simultaneous tracking of features. State-based reasoning was implemented in the trend recognition and the prognostic modules of a prototype Space Station Freedom Maintenance and Diagnostic System (SSFMDS) developed at Honeywell's Systems and Research Center.

2.0 SPACE STATION APPLICATION

The Space Station Freedom Maintenance and Diagnosis System (SSFMDS) project was established as a feasibility study whose purpose was to illustrate how Expert Systems could augment the Fault Detection, Isolation, and Recovery (FDIR) of Freedom's Attitude Determination and Control System (ADCS). The ADCS comprises four subsystems, called Orbital Replaceable Units (ORU), three of which are Honeywell's responsibility: the Star Trackers (ST) and Inertial Sensor Assemblies (ISA) for attitude determination and the Control Moment Gyroscopes (CMG) for attitude control. (The other subsystem is called the Reaction Control System and is a set of hydrazine jets also used for attitude control.)

SSFMDS comprises a set of two cooperating expert systems, one running on Freedom and the other running either on Freedom or on the ground. Since the onboard expert system is in closer contact with the environmental sensors, we call it the "On-line" system. The other, of course, is called the "Off-line" system and receives its information through the On-line system. When running in the actual Freedom environment, the two expert systems will communicate over the main telemetry link.

In our prototype, we demonstrate the operation of the On-line and Off-line systems by running each on a separate machine. We have the ability to simulate the telemetry link using either a serial RS232 or an ethernet connection. The prototype currently runs in a Unix/ XWindows/Common Lisp environment and supports both IBM PS/2 Model 70 portables and Sun workstations in any combination. One of the reasons that we chose the IBM as a prototype host is that it contains the processor (the Intel 80386) that has been chosen to be the heart of Freedom's Standard Data Processor (SDP).

An area that we do not cover in detail in this paper but that nevertheless deserves some mention is that of data filtering techniques. We have extended methods described in [Washington and Hayes-Roth, 1989] to significantly reduce the bandwidth required to communicate health monitoring and other ADCS data between the On-line and Off-line systems. Traditionally, bandwidth on spaceport telemetry links has been very tight -- there is never enough to go around and Space Station Freedom is no exception. It uses these techniques (including, for instance, dynamic thresholding) to send only the necessary data between the On-line and Off-line systems.

SSFMDS has three areas of expertise: predictive maintenance, diagnosis, and maintenance aiding. Of the three, the first has received the least attention in the more traditional Space Station Freedom software and thus has been the area where we have concentrated most of our efforts. Techniques we have developed for predictive maintenance, including trend recognition and failure prediction, form the main topic of this paper.

2.1 Why not use model-based reasoning?

A word about model-based reasoning: although we have built a number of expert systems, many of which have been model-based, we found difficulty in applying model-based techniques to the domain of predictive maintenance on Freedom. One problem is that there are conditions we predict which have no known physical model. One example is the degradation of the mirror in the ISA lasers. There is a relationship between the current/ output power curves such that prediction of a laser failure is possible several months away. No one knows why this relationship exists, which to say the least, makes it difficult to model. Cases like this have caused us to look for other methods.

3.0 STATE-BASED FEATURE RECOGNITON

One method we found, and the subject of this paper, is to represent trends and predictive scenarios in the ADCS as state machines. This technique, called State-Based Feature Recognition (SBFR), will be discussed in the following paragraphs.

3.1 Feature Representation

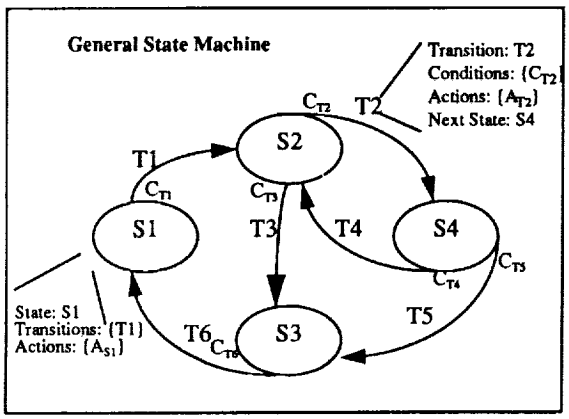


Figure 1 Sample State Machine.

Features to be recognized using SBFR are represented as finite state machines, with each feature having its own state machine. The following refers to Figure 1 which illustrates an example of a generic state machine. The state machine is made up of a set of states, {S1,...,S6} and transitions between those states, {T1...T6}. Each state in the machine represents a stage in the identification of the feature. Each state has associated with it a set of transitions and, optionally, an action to be executed when the state is entered, called a state action. Similarly, each transition may, optionally, have an action associated with it, called a transition action. State actions are useful if entering a state means something no matter how it was entered. For example, suppose that entering state S3 implied a dangerous engine condition, a state action would handle this no matter how the state was entered. Transition actions, on the other hand, are useful in situations where entering a state means something different depending on the transition traversed. For example, in Figure 1, a notification may necessary when entering state S3 from state S2, while no action is needed when entering state S3 from state S4.

Each transition has a set of conditions associated with it. The conditions define when the transition may be traversed. In other words, when the condition for a particular transition evaluates to true, the machine can traverse the transition. For example, in Figure 1, C_{T2} is the condition for T2, if the current state is S2 and C_{T2} becomes true, S4 will become the next state. When that occurs, the actions defined for T2, $\{A_{T2}\}$ will be executed. If S4 as any state actions associated with it they will also be executed.

The machines described above have their roots in automata theory. State machines that specify actions on the transitions are called Mealy machines and those whose actions are specified on the states are called Moore machines. From automata theory, any Moore machine has an equivalent Mealy machine and, conversely, any Mealy machine has an equivalent Moore machine [Hopcroft and Ullman, 1979]. Thus, only one type of action is required. Transition actions can be replaced by adding more states and associating actions with those, while state actions can be replaced by defining the action on all of the transitions entering the state. Eliminating either type of action, however, has certain drawbacks when used in SBFR. A transition action, replaced by a state action, increases the complexity of the machine by adding more states to the machine, while a state action replaced by transition actions will make the machine harder to maintain. That is, instead of maintaining one action for the state, actions must be maintained for all of the transitions. Consequently, supporting both types of actions is recommended to preserve the intuitive nature of the machines.

During operation, only one state of the machine is active, called the current state. The transitional conditions are, therefore, typically mutually exclusive to ensure that two state transitions cannot simultaneously be true. Of course, there may be circumstances when it is reasonable to have a machine be in two states at once, and state machines can be implemented to support that, but for the applications discussed below, it is more intuitive to limit the machine to one currently active state. This paper assumes that the transitions for given state are mutually exclusive and that there is only one currently active state per machine.

The semantic meaning of a feature is captured by the states and transitions, while the specifics of the feature (i.e. the exact data which causes the state machine to move from one state to another) are captured by the transitional conditions. This results in a separation between the general definition of a feature and its real world implementation, allowing a general machine to be

instantiated in many different contexts. This, in effect, permits the construction of a library of feature descriptions that can be instantiated for many different applications by only changing the specifics of the transitional conditions. For instance, an increasing trend in computer CPU utilization and daily air temperature could have the same general definition (i.e. the same states and transitions), but different transitional conditions. The CPU trend will have a much smaller time span and magnitude than the temperature trend. Since these trends are so similar, the state machines will be identical except of the magnitude and duration of the feature.

3.2 SBFR Application Characteristics

Each state machine moves from one state to another in a well-defined order that depends on which transitional conditions evaluate to true. Thus, features recognized by a well-defined order of stages are those best represented and recognized using SBFR. The stages are usually ordered temporally, although other orderings may also be possible. As will be discussed later, in section 4.0 and section 5.0, failure prediction and trend identification are features whose stages are temporally ordered.

One case where the features are typically not well-ordered is fault diagnosis. Faults are typically recognized by the presence of a set of symptoms that appear at the time of the failure. These symptoms usually do not appear in any well-defined order (if they did then a state machine could be used as a means to predict the failure), resulting in a state machine of only two states, essentially an if-then statement. So, while fault diagnosis could certainly be implemented using SBFR, another approach would probably be better.

3.3 SBFR Is Data-Driven

The fact that SBFR has a natural application to features recognized by a well-defined sequence of events implies that SBFR should execute in a data-driven manner. That is, the SBFR module is invoked when new data enters the system. The transitional conditions associated with the current state are evaluated using the new data and any required historical data. If a transition's condition is true, the machine will move to the next state, executing any actions defined on the transition and the new state entered.

The data-driven nature of state machines allows features to be detected concurrently. The same data can be used for any number of state machines, allowing a system implementing SBFR to simultaneously track several features in the incoming data. In many situations, particularly when real-time data analysis is required, the ability to track many features in parallel is not only useful but necessary.

The following sections will illustrate in greater detail how SBFR is implemented to recognize features in parameter data. Two implementations of SBFR are present in the SSFMDS, trend recognition and failure prediction. Both implementations are based on the general machines discussed above.

4.0 TREND RECOGNITION

One important implementation of SBFR is the difficult, but important, task of trend recognition. Trends can indicate many things about a particular device including impending failures, environmental changes, and other anomalies. It often takes an expert to decipher trends and speculate as to their meaning, especially since the significance of trends is dependent on many factors (e.g. the feature's time span, the parameter's expected behavior, etc.). For instance, fluctuating data for a generally stable parameter would indicate a problem, while the same data for an erratic parameter would be considered normal behavior. Humans are very good at analysis involving pattern detection in noisy, convoluted data. Domain experts are even better at this sort of analysis, since they know what parameter trends are significant to the monitored device. Experts are, however, a scarce commodity and their time is too valuable to watch data scroll by on a screen. Furthermore, some trends may occur too rapidly for a human to detect it, some significant trends may happen in milliseconds. These two problems have lead us to search for ways to automate the process using computers. The problem, however, is programming the computer to both recognize significant parameter trends and ignore the insignificant ones. SBFR provides a representation scheme encompassing an intuitive method for describing the trend in human terms and an easy way to translate it into computer terms.

4.1 SSFMDS Implementation

The state machines implemented to recognize trends are cleverly called trend machines. Each trend machine is built on the principles discussed in section 3.0, with the addition of an initial state and a final state. The initial state is the default starting state of the machine and represents no trend being detected, i.e. no evidence of the trend has been seen. The final state is just the opposite, entering this state indicates that a trend has been identified. Like the general state machines, a trend machine will remain in a state until incoming data causes it to transition to the next state.

The final state, denoted by a double circle in Figure 2 and Figure 3, is an important state of the trend machine. When the machine enters this state, a trend has been detected and a notification is sent to the knowledge base, along with the information collected as the trend was being detected. As long as the machine remains in the final state, the trend notification is updated with the current information, allowing the SSFMDS to monitor long-term trends with a minimal amount of effort.

A set of trend machines is associated with each parameter and uses data local to that parameter as input. A global view encompassing the entire ADCS could be provided, but would dilute the intended function, which is to provide information about ORU parameters to the SSFMDS reasoning mechanisms at a higher level than raw data. Features dependent on several ORU parameters do exist and are considered in the discussion of failure prediction, see section 5.0.

Since trend machines are associated with a single parameter, they can be executed efficiently. When new data enters the SSFMDS, the only transition condition functions evaluated are those associated with the current states of the parameter's machines. A global view would require evaluating the transition functions of all trend machines. This can be significant when the system being monitored has several hundred parameters, each of which has several trends identified for it.

A parameter's trend machines are implemented hierarchically. Filtered data from the on-line system is input into the first trend machine layer. The machines at this level recognize simple trends like increases, decreases, spikes, etc. Trends of this nature only require raw data as input. These trends are used as input into the next layer of trend machines which can recognize more complex trends. Currently, the SSFMDS trend recognition module implements the first layer of the hierarchy and has the machinery to implement higher layers. This hierarchy enables complex trends to be broken down into components consisting of simpler trends. The following two examples will illustrate trend machines at the first and second level of the hierarchy.

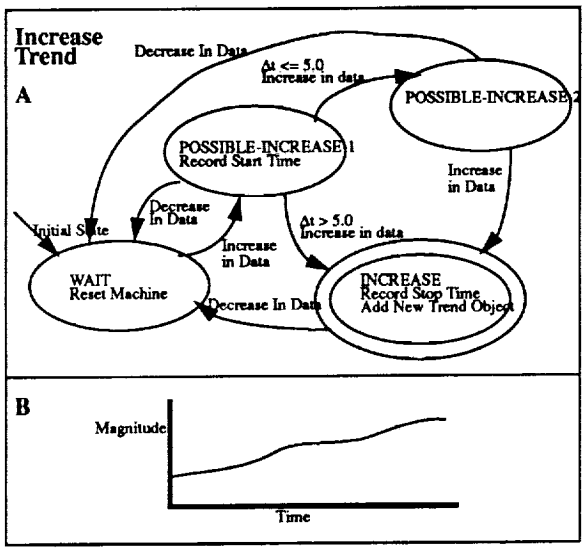


Figure 2 Simple Increase Trend and Sample Data

Figure 2 A shows a trend machine used to recognize a simple increase trend, like the one illustrated in Figure 2 B. Note that this is just one example, the transitions and states defined for this machine could be changed to suit any situation. In this case, an increase is defined as being two jumps in the data occurring greater than 5 time units apart, or three jumps in the data if the first two jumps happen in less than or equal to 5 time units. The reason for the time limit is to differentiate between an increase and a spike. Anytime a decrease in the data is seen, the machine will transition back to the initial state. When the machine enters the final state (denoted by a double circle in the diagram) the knowledge base is informed that an increasing trend has been identified and is sent the information collected during the trend identification (the start and stop times in this case).

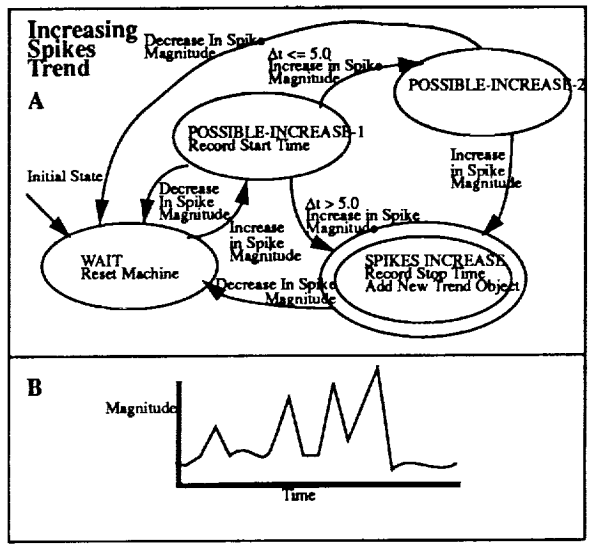


Figure 3 Increasing Spikes Trend and Sample Data.

Figure 3 shows a trend machine defined at the second level of the hierarchy. In this case, the machine is used to recognize an increasing-spikes trend, i.e. a trend of spikes with increasing magnitude, like the one in Figure 3 B. This trend is at the second level of the hierarchy because it takes simple trends (spikes in this case) as input and outputs trends made up of those simple trends. When a spike is noticed by the trend recognition module, it is used as input into the machines at the next level where the spike magnitude is used in the transitional conditions. Notice that the only difference between the machines of Figure 2 and Figure 3 is that the transitional conditions are different, the states are otherwise identical. In Figure 2 the conditions use the value of the incoming filtered data, but in Figure 3 the magnitude of detected spike trends is used. This illustrates how one basic trend machine can be applied in different circumstances. The trends detected provide information to personnel monitoring the ADCS as well as information to other modules of the SSFMDS, specifically the failure prediction module.

5.0 FAILURE PREDICTION

Predicting when a failure will occur is a critical factor in the maintenance of the ADCS. The ADCS is characterized by functionally independent ORUs, low failure rates, and extremely high replacement and repair costs. Spare parts not stored onboard the station must be flown up by space shuttle or rocket, and, regardless of how the replacement arrived, EVA activity is currently required to replace a failed ADCS ORU. Either of these activities can cost millions of dollars and puts equipment and, more importantly, personnel in danger. Forecasting ORU failures can yield tremendous savings by allowing EVA times to be scheduled to accomplish several tasks and allowing spare parts to be flown up on scheduled flights.

Predicting failures, however, is not a simple task and a general approach is not yet available. Failure prediction relys heavily on the experience of experts and their ability to estimate the condition of the equipment. During our research, we found that experts predict failures by noticing certain features in the data over a period of time. For example, features in the ISA laser output current/power can indicate a failure several months before it occurs, allowing plenty of time for repairs to be made. A state machine can be used to capture this information.

5.1 SSFMDS Implementation

The technique is very similar to that used for identifying data trends, discussed in section 4.0. A state machine is defined for each predictive scenario. As data is made available, it is fed into the state machine. The actions associated with the states and transitions are cautions and warnings indicating the estimated time to failure and recommended actions. The data used by the predictive machines includes raw data and trend information from all parts of the ADCS. That is, a predictive machine may monitor trends across different parameters on one or more ORUs. This differs from the approach taken for trend analysis that only considers data for one ORU. For instance, one predictive machine can monitor trends in the wheel unbalance of all ADCS CMGs. These trends should be roughly equal, if one CMG exhibits a different trend (i.e. it is increasing while the others remain steady) a problem may be indicated.

To make this more concrete, consider an example of a CMG predictive scenario that illustrates how a predictive state machine can combine trends from multiple parameters to predict a failure. This example, a CMG spin bearing failure, is based on a failure that occurred on Skylab, rendering one of its CMGs inoperable. This type of spin bearing failure is characterized by a series of zero or more spin motor current spikes followed by a rise in the spin motor current, shortly after this rise, the spin bearing temperature also increases. If this continues, the CMG wheel will seize and stop. The sequence of symptoms is important, if the same sequence were seen in a different order it may indicate a different problem, or no problem at all. If a different problem were indicated, a separate state machine would have to be created to monitor that problem in parallel with the bearing failure machine.

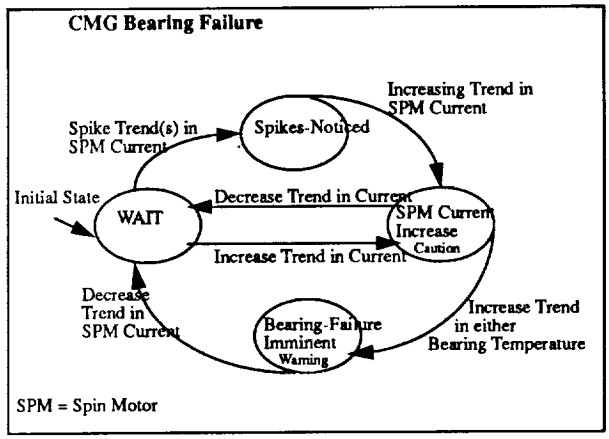


Figure 4 Example Predictive State Machine

The state machine reflecting this behavior is shown in Figure 4. After reading the section on trend analysis, the machine should be fairly self-explanatory. As data enters the system and CMG trends are detected, they are fed into the state machine. For instance, if the machine is currently in the WAIT state, the machine is monitoring the data for either spin motor current spikes, or a spin motor current increase trend. When this happens, it will transition to the next state, executing whatever actions are defined on the transitions.

In this case, if an increase was seen in the spin motor current, a caution would be issued by the system indicating that a bearing failure may occur. Notice that the detection of current spikes does not trigger an action. Current spikes were detected prior to the Skylab failure, but were also detected at different times on other Skylab CMGs that did not fail. The presence of spikes, therefore, would reinforce any subsequent conclusions but are not, themselves, indicative of a bearing failure. In other words, current spikes are used to increase confidence in the prediction, but a prediction is not based solely upon their presence.

6.0 FUTURE DIRECTIONS

This research could be extended in a number of directions.

- Embed the state machines in the ORU's and the sensors. Putting the intelligence at the sensor or the ORU would reduce the bandwidth required to report on problems and potential problems to a trickle, freeing it up for other data.One particular avenue we are exploring in this area is making sensors intelligent by providing them with data filters and trend recognition [Wald, Schoess, and Hadden, 1991].

- Explore the relationship between the data filters and the trend machines. It may be the case that ways can be found in which the two subsystems can cooperate to provide an even higher data rate and/or lower bandwidth than they do now.
- Add higher level trend machines. Making the output of one trend machine available as the input to another one turned out to be a good idea, yet no more than two layers have been tested. Perhaps the logical extension of this idea into more layers of trend machines would have application.
- Implement the machines in a forward chaining rule-based language like CLIPS. CLIPS (both the original and CLIPS-Ada) is emerging as a NASA standard. This, coupled with the facts that a forward chaining language is well-suited to a state-based system and the ease of extensibility such a language affords, make it an attractive choice for an implementation vehicle.
- Learning. A number of trends are currently recognized, yet specification of a complete set of "interesting" trends in any one domain is a difficult task. Machine learning techniques may allow us to generate the ability to recognize these trends with a minimum of human input. A more speculative idea is to use neural networks to perform this learning.

Finally, perhaps the most important future direction is the application of the Maintenance and Diagnosis System to other space related projects. It is clear that these systems must be many times more intelligent than their earth-bound counterparts -- a vessel halfway between the Moon and Mars cannot call a tow truck. At the same time, power and space requirements dictate their own constraints. We must use existing technology and develop new technology to assure that our astronauts' journeys are safe.

7.0 Bibliography

Hopcroft, J. and Ullman, J.
Introduction to Automata Theory, Languages, and Computation
Addison-Wesley, 1979.

Wald, J., Schoess, J. and Hadden, G.
Distributed Health Management Systems for GN&C Applications
1st International Conference for Guidance, Navigation, and Control, 1991
Noordwijk, The Netherlands.

Washington, R. and Hayes-Roth, B.
Input Data Management in Real-Time AI Systems
International Joint Conference on Artificial Intelligence, IJCAI-11
1989 pp. 250-255.

N93-11926

REQUIREMENTS AND OPPORTUNITIES FOR SATELLITE AUTONOMY

Mr. W. Spencer Campbell, The Aerospace Corporation, Alb NM
Capt Chris Gdanski, USAF, Phillips Laboratory, KAFB, NM

ABSTRACT

The requirements for autonomous control of spacecraft functions have been well articulated by a wide selection of US Air Force mission analyses and architecture studies. The opportunities, however, to implement these autonomy functions have been limited. An overview of documented requirements is presented to demonstrate the scope of autonomous functions needed, and potential time frames for implementation is derived. The missed opportunities are analyzed to provide lessons learned, and near term opportunities will be reviewed. The current initiatives at the Phillips Laboratory are described in terms of the requirements addressed and the proposed approaches will show program risk reduction for satellite autonomy technology insertion into space operations.

Session I2: DIAGNOSIS

Session Chair: Tim Hill

N93-11927

AN EXPERT SYSTEM FOR DIAGNOSING ENVIRONMENTALLY INDUCED SPACECRAFT ANOMALIES

by

Mark Rolincik, University Research Foundation, Greenbelt, MD 20770
Michael Lauriente, NASA Goddard Space Flight Center, Code 410, Greenbelt, MD 20771
Harry C. Koons & David Gorney, The Aerospace Corporation, M-2/260, P.O. Box 92957, Los Angeles, CA 90009

ABSTRACT

We are designing a new rule-based, machine independent analytical tool for diagnosing spacecraft anomalies using an expert system. Expert systems provide an effective method for saving knowledge, allow computers to sift through large amounts of data pinpointing significant parts, and most importantly, use heuristics in addition to algorithms, which allow approximate reasoning and inference and the ability to attack problems not rigidly defined.

The knowledge base consists of over two-hundred (200) rules and provides links to historical and environmental databases. The environmental causes considered are bulk charging, single event upsets (SEU), surface charging, and total radiation dose.

The system's driver translates forward chaining rules into a backward chaining sequence, prompting the user for information pertinent to the causes considered. The use of heuristics frees the user from searching through large amounts of irrelevant information and allows the user to input partial information (varying degrees of confidence in an answer) or 'unknown' to any question.

The modularity of the expert system allows for easy updates and modifications. It not only provides scientists with needed risk analysis and confidence not found in algorithmic programs, but is also an effective learning tool, and the window implementation makes it very easy to use. The system currently runs on a MicroVAX II at Goddard Space Flight Center (GSFC). The inference engine used is NASA's C Language Integrated Production System (CLIPS).

BACKGROUND

This joint project to develop an expert system for diagnosing environmentally induced spacecraft anomalies owes its origin to the observation by the Air Force that spacecraft anomalies were often environmentally induced. It was a common occurrence then to receive calls on problems with satellite anomalies. The expert system was initiated as a research project that could be used by program offices and contractors to eliminate the flood of calls which were becoming a nuisance. The objective was to develop a classic diagnostic tool for trying to determine, once an anomaly has occurred, whether it was caused by the environment. Hopefully this information will be useful to the design of spacecraft, so that in the future, systems will be built having increased immunity to the hostile space environment.

Historically, the Air Force has supported NASA's EnviroNET (1), so its on-line feature was considered a "natural" as a communication tool for educating its users about this innovative venture. In addition, an opportunity existed for users to feed back information that might improve on the system. The key to advancement in this endeavor is communication between users. The user here is either a forecaster, a scientist, an engineer, an operator, or perhaps a contractor who needs to know something about the effects of the environment on a satellite or a satellite subsystem, recognizing that they will have access to a variety of databases and knowledge. A special session on environmentally induced spacecraft anomalies chaired jointly by the Air Force and NASA at the 1990 AIAA meeting in Reno (2) brought to focus the issues of concern.

INITIAL WORK: METHODOLOGY

This research tool was originally developed by the Aerospace Corporation for a PC using a Texas Instrument commercial expert shell (3). It was then handed off to EnviroNET to develop a program to port the system to the EnviroNET central computer, which is accessible through most of the popular networks. The inference engine used is NASA's C Language Integrated Production System: CLIPS (4). CLIPS is not only compatible with both C and Fortran languages, but it has features which include the ability to compile the rules and save them in a binary image file, thus allowing faster execution than a typical rule interpretive system. This feature qualifies CLIPS to be used as an expert shell, i.e., an environment where the rules can reside and be accessed.

Initially, recognized experts in the field were queried on how to diagnose anomalies; these "rules of thumb" were formatted into logical rules. The expert system rules involve four main types of environmental anomalies: bulk charging, surface charging, single event upsets (SEU), and total radiation dose. It should be recognized that the expert system as a tool can be expanded to include other causes of anomalies, even non-environmentally induced anomalies.

The architecture of the system was designed to emulate the way the user normally looks at data to diagnose anomalies. The expert system not only consolidates expertise in a uniform, objective, and logical way, but it also offers "smart" ways of accessing various databases which are transparent to the user. By applying various rules in its knowledge base, the system accesses databases, queries the user as appropriate, and arrives at a conclusion. The system output was verified by referring to historical case studies and historical data.

EXPERT RULES

The EnviroNET expert system knowledge base currently contains over two hundred (200) rules. The system goes through a varying "decision tree" based on these rules and user input in order to arrive at the likely cause of an anomaly. The rule base includes the expert system rules in a defined "if-then" format that will be "fired" under the control of the inference engine. The user interface links to databases which include past environmental data, satellite data, and previous, known anomalies. Information regarding satellite design, specifications and orbital history needs to be assimilated with previous anomalies data and environmental conditions, while addressing the specific circumstances of individual users.

Seldom are the environmental problems encountered by scientists rigidly defined, and thus they lack clear mathematical solutions. Under such circumstances, algorithmic programs are too limited by their sequential logic, becoming too cumbersome when trying to consider a wide range of variables of varying degrees of certainty.

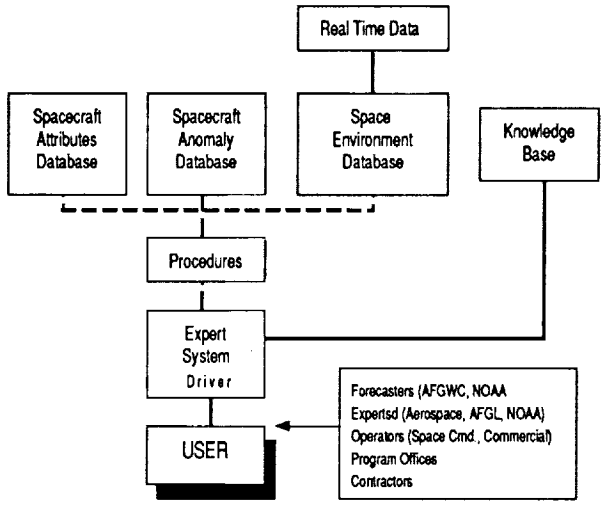


Figure 1. Expert System Configuration

EnviroNET's Expert System is being developed as an Artificial Intelligence (AI) technique to cope with this voluminous data and fluidity associated with spacecraft/environment causality models.

Unlike its algorithmic predecessors, an expert system can be flexible in the way that it attacks complex problems. By virtue of its three basic parts (a knowledge base, a fact base, and a driver), an expert system more closely simulates the methods of human experts who use a combination of known, empirically derived formulae, hunches based on degrees of certainty and experience, and even judicious "fudging" when specific data is lacking. Figure 1 shows the expert system configuration.

The knowledge base, with its set of rules, is what makes a rule-based expert system unique. Best thought of as an independent collection of "if...then" statements, the rules are created by experts in their respective fields and reflect the current level of human experience, along with its uncertainties. Under the weight of these rules, and by the use of multi-field variables, an expert system can be said to "ponder the possibilities" presented by a compendium of data and knowledge too extensive to be readily assimilated by any single person. Rather than being limited to conclusions that must satisfy a set of tightly ordered mathematical statements, the system is free to offer suggestions, considerations, and likelihoods.

```
RULE201
=======
  SUBJECT :: BULK_CHARGING-RULES
  DESCRIPTION :: (recurs when fluence high)
If  1) the recurrence of the anomaly, and
    2) the recurrence is OF_HIGH_PENETRATING_FLUX, and
    3)  1) the seven-day accumulated fluence of penetrating electrons is
           HIGH, or
        2) the seven-day accumulated fluence of penetrating electrons is
           VERY_HIGH,
Then there is suggestive evidence (60%) that the cause of the anomaly is
BULK_CHARGING.

    IF   :: (RECURRENCE AND PERIODICITY = OF_HIGH_PENETRATING_FLUX AND
             (ACCUM_FLUEN = HIGH OR ACCUM_FLUEN = VERY_HIGH ) )
    THEN :: (CAUSE = BULK_CHARGING CF 60)


RULE110
=======
  SUBJECT :: TOTAL_DOSE-RULES
  DESCRIPTION :: (Local time recurrence rules out total radiation dose.
If  1) the recurrence of the anomaly, and
    2) the recurrence of an anomaly in a specific local-time sector.,
Then it is definite (100%) that the cause of the anomaly is not TOTAL_DOSE.

    IF   :: (RECURRENCE AND LT_RECUR)
    THEN :: (CAUSE != TOTAL_DOSE)
```

Figure 2. Rule Format

```
INCLINATION
===========
  TRANSLATION :: (the inclination of the plane of the orbit with respect
                  to the earth's equatorial plane )
  PROMPT :: (Select the inclination of the satellite with respect to the
             earth's equatorial plane. )
  TYPE :: SINGLEVALUED
  EXPECT :: (EQUATORIAL LOW_INCLINATION HIGH_INCLINATION POLAR OTHER)
  UPDATED-BY :: (RULE041 RULE133 RULE134 RULE135 RULE136 RULE132 RULE138
                 RULE139 RULE140 RULE141 RULE142 RULE137 )
  ANTECEDENT-BY :: (RULE026 RULE030)
  USED-BY :: (RULE017 RULE016 RULE091 RULE089)
  HELP :: ("Low inclination orbits are below 30 deg.  High
            inclination orbits are above 60 deg.  Polar orbits
            are above 80 deg.  Interplanetary orbits are undefined." )
  CERTAINTY-FACTOR-RANGE :: UNKNOWN


LT_RECUR
========
  TRANSLATION :: (the recurrence of an anomaly in a specific local-time
                  sector. )
  PROMPT :: ("Indicate the degree of certainty that you have that this
              type of anomaly has a strong tendency to recur in one local
              time sector, for example the nightside or the dayside of the
              earth?" )
  TYPE :: YES/NO
  USED-BY :: (RULE019 RULE020 RULE110 RULE054 RULE188 RULE189 RULE190
              RULE191 RULE192 RULE193 RULE194 RULE043 )
  HELP :: (The anomaly should have occurred a few times (i.e. six or more)
           before you have confidence that the recurrence is related to a
           specific local-time sector.  Generally we are asking if the
           anomaly has a very strong tendency to occur within a 12 hour range
           in local time. )
  CERTAINTY-FACTOR-RANGE :: POSITIVE
```

Figure 3. Variable Format

The rule format used in the expert system is shown in Figure 2. Each rule has a subject associated with it (in this case one of the four causes considered), a description of the rule, and then the actual rule itself. The rules also have what is termed a 'confidence factor' associated with the right hand side of each rule. Algorithms, which normal programs are limited to using, have a 100% certainty to them and are a subset of the general heuristic rules which the expert system uses.

This aspect of the rule-based expert system is very important in diagnosing anomalous behavior since much of the knowledge, rules and experience required to diagnose these anomalies have confidence factors associated with them. The use of such confidence factors in the expert system introduces the concept of 'risk assessment' to the diagnostic procedure and the inclusion of knowledge which otherwise would be lost, since it is, at the very least, extremely difficult to represent such knowledge using mathematical formulae.

Another advantage of using a rule-based system is that it allows direct access to and easy comprehension of the knowledge and expertise used to diagnose the anomalies as opposed to the very complicated, and sometimes esoteric coding of most normal programs. Not only does this provide a way of storing the knowledge, it also allows the system to be easily and quickly updated. These updates are accomplished by simply adding, deleting or modifying rules to which the system then automatically adjusts.

VARIABLES

The EnviroNET expert system's use of variables is another area which makes this system unique, allowing it to handle non-algorithmic, equivocal problems. A variable in this system can take on one of three settings. It can be '*unset,*' meaning that it has not been input by the user and that no rule has been able to determine a value for it; it can be '*unknown,*' which means the user was prompted for the variable but did not know it; or it can have one or more '*values.*' The unique aspects of the system are that not only can the expert system continue to execute when variables are unknown, but when variables do have values, each value has a confidence factor associated with it. Figure 3 shows examples of variable formats.

In the variable format, the translation and prompt string are self-explanatory. Each variable also has a type associated with it, either 'single-valued,' 'multi-valued,' or 'yes/no.' The 'expect' field is a list of the possible values for that variable which the user can select when and if he/she is prompted for that variable. The 'updated_by' field is a list of rules which are able to determine values for that variable, while the 'used_by' field contains rules which require this variable in order to fire. (It is possible that in order for a rule to fire, a variable must be '*unknown.*') The 'help' field is the information displayed when the user presses the help key, requesting more information on the variable being prompted for. The 'certainty-factor-range' (CFR) is particular to this system and can have a value of '*unknown,*' '*positive,*' or '*full.*' The CFR being '*unknown*' means that this is a possible input for that variable. If the CFR is '*positive,*' the user can input degrees of confidence from 0 to 100 for each of the entered values for that variable. Finally, if the CFR is '*full,*' the user can input degrees of confidence from -100 to 100, which means a range from

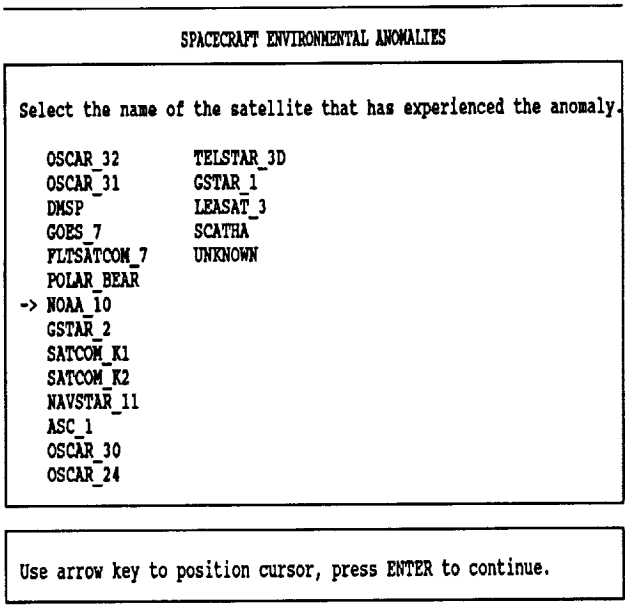


Figure 4. Satellite Selection

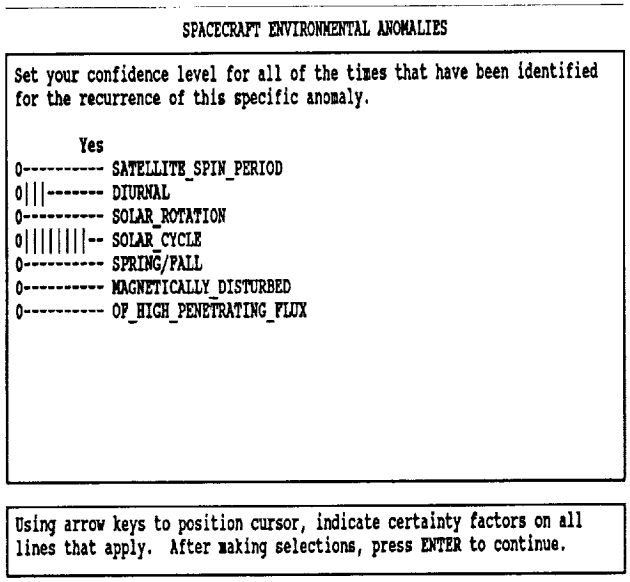


Figure 5. Multi-valued input with confidence

being 100% certain the variable is *not* a specific value to being 100% certain that the variable *is* a specific value.

The confidence factors relay the confidence the user has in a certain value of the variable. This is very important since there is most likely information of which the user is not 100% sure. Such information is lost in normal programs. The combination of the confidence factors of variables and those of the rules propagates the confidence factors to other variables which are determined by these rules and ultimately to the cause of the anomaly.

Figure 4 shows an input screen for a single-valued

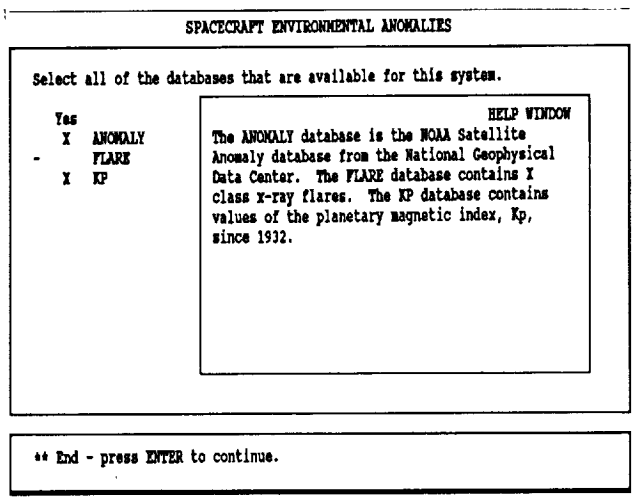


Figure 6. Database selection screen

variable which assumes 100% confidence and a CFR of '*unknown.*' Formats like this figure are examples of the user friendly interface that was designed and are intended to portray a snapshot image of what the user sees.

Figure 5 is an example of the input screen for a multi-valued variable with a '*positive*' CFR. Notice how the variable in Figure 5 can have more than one value, and each value has its own confidence factor associated with it.

FACT BASE

The fact base, a collection of informative sources related to the topic of interest, is the second integral part of an expert system. It can consist of as many separate data bases as may be deemed pertinent to solving the problem at hand. In the case of spacecraft anomalies, a fact base might contain information on the hardware currently in use, other active and past satellite systems, and historical data for orbital environments.

The database screen is shown in Figure 6, which shows the databases available for this system along with an example of the expert system help facility which is available for any variable. An important advantage obtained in using the expert system is that once it has been established which databases are available, the rules determine which information is pertinent, access the database for the relevant information and apply this information (all of which is transparent to the user). Also, the database accessing is modular and easily expandable, thus if more databases need to be added, only the selection screen needs to be changed and the new rules added to the knowledge base. These capabilities free the user from sifting through large amounts of data and ensure that only pertinent information and all pertinent information is used in the diagnosis.

INTERFACE

The interface is one of the aspects which makes all expert systems different from one another. Since the expert shell, databases and knowledge base are independent and modular, the main purpose of the interface is to create a coordinating system which is not only user friendly, but also provides the necessary features to assist the user in understanding the system and the results.

The system's current interface driver translates forward chaining rules into a backward chaining sequence, prompting the user for information pertinent to the causes he/she wishes to consider. The main purpose of the driver is to maintain information regarding the variables which are being determined, the rules which can determine these variables, the status of the variables, and which rules can be fired.

Some variables are designated as initial or goal variables. The system first prompts the user for the initial variables. The driver then stacks the goal variables on the run time stack and searches the knowledge base for rules which determine (or 'update') these variables, and then puts them on the stack as well. The system focuses on those possibilities of high confidence and then assists the user by directing him/her to areas of consideration that directly affect the particular problem. The goal (variable) in our system is the CAUSE of the anomaly, a multi-valued field variable with a *'full'* CFR, since it can take on any number of the four possible causes where each cause has its own confidence factor associated with it ranging from -100 to 100.

If a variable on the left hand side of a stacked rule is unset, this variable becomes the current goal variable and is put on the stack, and the process continues. If a variable is on the stack and has not been determined by any rules nor by the available database (and it has a prompt string), the user is prompted for it. This can be thought of as a transformation of the forward chaining rules in the knowledge base into a backward chaining variable sequence. Once a variable has a value, it is removed from the stack and the rules which use this variable are fired, discarded, or require the driver to put the next variable on that rule's left hand side onto the stack. The chaining process continues until the stack is empty.

Any rule on the stack that can be fired does so transparently to the user, where the confidence factors of the individual variables on its left hand side (LHS) are used for determining the confidence or validity of the entire LHS. When a rule fires, it executes the right hand side (RHS), and the confidence factor associated with its LHS is used in conjunction with the confidence factor of the rule to propagate the confidence to the RHS. This RHS execution can entail the setting of variables, the use of mathematical calculations, or the accessing of databases.

LEARNING TOOL

One of the most beneficial aspects of the system is its use as a learning tool for diagnosing spacecraft anomalies. A user is initially given a choice between either 'novice' or 'expert' mode for the current session. If the user selects the novice mode the system automatically gives detailed explanations and descriptions of terms and reasoning as the session progresses, in a sense teaching the user about the topic or topics. The expert mode, on the other hand, simply executes the session without giving these extra explanations, unless the user specifically requests them.

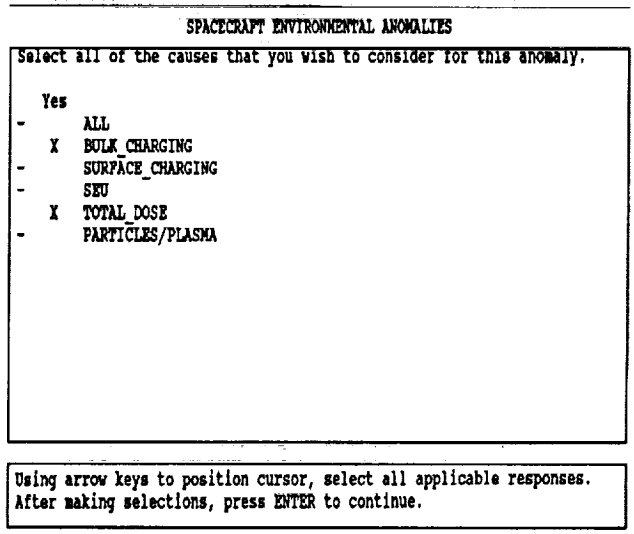


Figure 7. Causes selection screen

The user is also given the option of selecting which causes are to be considered. (See Figure 7.) This selection determines a knowledge base sub-group, so that only rules in this specific environmental area are considered. In this way the user can learn what variables, information and data affect and are important to that cause. In addition to this, in the features described next, the user is actually able to access the relevant rules him/herself and other variables and facts which were determined by using these rules.

UNIQUE FEATURES

The ability to add intricate features and options is primarily due to the modularity of the system which the expert shell and expert system knowledge base concept

itself provide. These features are the most impressive when demonstrating the capabilities of the EnviroNET expert system and its advantages over the usual, strictly mathematical programming techniques.

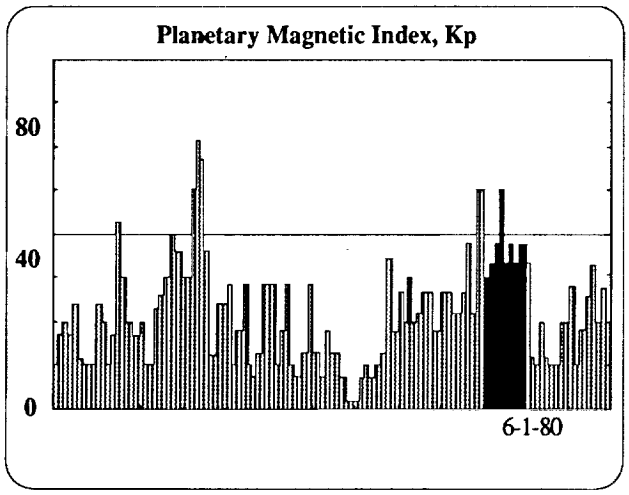


Figure 8. Kp Graph

```
SPACECRAFT ENVIRONMENTAL ANOMALIES

Setting  TOTAL_DOSE_TECHNOLOGY  =  AMORPHOUS_TTL  cf  100
Testing  RULE119
RULE119  FAILS
Testing  RULE128
RULE128  FAILS
Testing  RULE129
RULE129  FAILS
Testing  RULE131
Applying  RULE131
Setting  TOTAL_DOSE_THRESHOLD  =  1000000  cf  100
Testing  RULE120
Applying  RULE120
Setting  CAUSE  =  TOTAL_DOSE  cf  -86
old cf  -30
Mark_antec_rules_for  CAUSE  RULE027
Try_marked_antec_rules
Testing  RULE027

** More - press ENTER to continue.
```

Figure 9. Trace example

The user interface provides for accessing graphics. For example, if the user inputs that one of the databases available is Kp, the system will ask if he/she wishes to see the Kp historical graph for the time around which the anomaly occurred. If the input is '*yes*,' then a graph similar to the one shown in Figure 8 will be displayed. (If, however, the date is '*unset*,' then the system will first ask for it, and if the date is '*unknown*,' the system will ignore this line of questioning altogether.) This gives the user a much needed overall view of environmental information and conditions around the date in question.

Another feature which makes the expert system unique is its trace capability. The user can turn on the trace and send it to the screen or a file. The trace shows the rules as they are tested, variables as they are pushed onto the runtime stack and determined, and searches of the databases (see Figure 9).

This allows the user to understand what is happening at any step and see the knowledge that is being used, thus giving the user confidence in the system. This type of capability is obviously not available in purely algorithmic programs. Due to the amount of information the user could be prompted for and depending on the particular session, the user may want to review his/her inputs. This capability is available in the 'REVIEW' facility. This option also provides the user with a simple way of comparing different inputs of different sessions.

```
SPACECRAFT ENVIRONMENTAL ANOMALIES

Enter a value between 0 and 400 for the maximum value of the planetary

                                                        WHY WINDOW

the three hour planetary index Ap is needed to determine the level of
magnetic activity in the magnetosphere

RULE094
If   the three hour planetary index Ap is greater than 30,
Then it is definite (100%) that the level of magnetic activity in the
magnetosphere is DISTURBED.

** More - press ENTER to continue.
```

Figure 10. Backward reasoning

```
SPACECRAFT ENVIRONMENTAL ANOMALIES

Enter a value between 0 and 400 for the maximum value of the planetary

                                                        WHY WINDOW

the level of magnetic activity in the magnetosphere is needed to
determine the cause of the anomaly

RULE021
If   the level of magnetic activity in the magnetosphere is QUIET,
Then there is suggestive evidence (50%) that the cause of the anomaly
is not BULK_CHARGING.

** More - press ENTER to continue.
```

Figure 11. Backward reasoning (con't.)

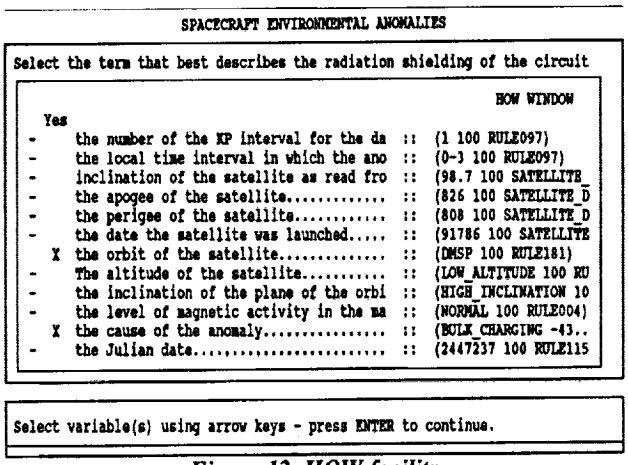


Figure 12. HOW facility

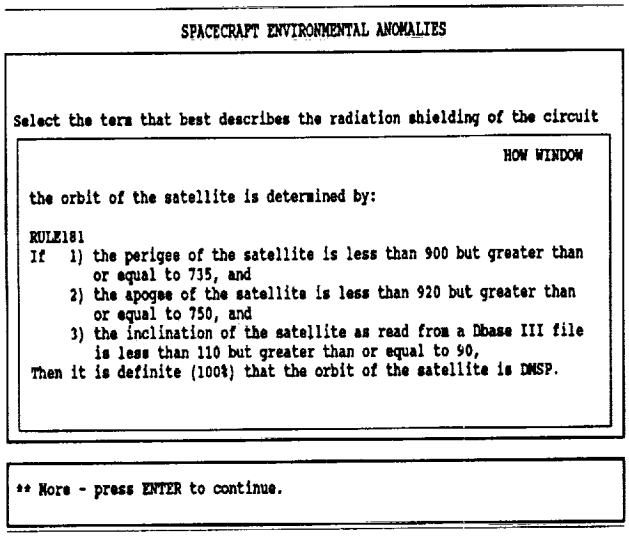


Figure 13. HOW facility (con't.)

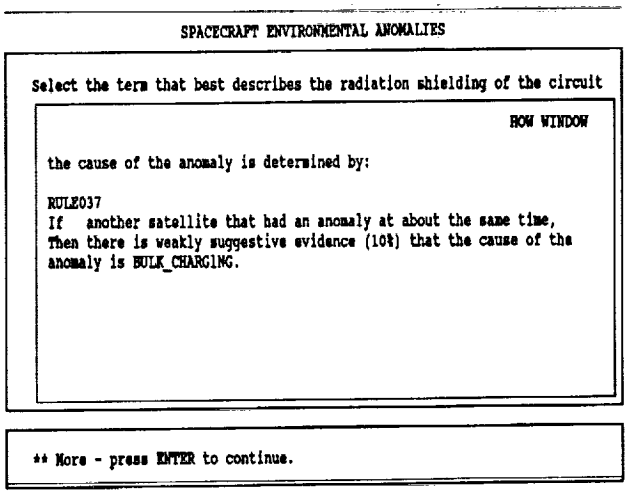


Figure 14. HOW facility (con't.)

A feature which demonstrates a definite advantage of the rule-based expert system is the 'WHY' option. Any time the system prompts the user for a variable, the user can ask the expert system why the system needs this variable. The system then uses its run time stack (a backward chaining stack) to follow and show the reasoning backward to the goal, that is, the cause of the anomaly. Figures 10 and 11 show an example of this. This is not only vital to understanding and having confidence in the system, but it is also an important part of the expert system's use as a learning tool.

A final feature which sets the expert system apart is the 'HOW' command. As with all programs, the expert system is constantly determining variables by means other than the user inputting them, whether by the heuristics and algorithms in the rules or by extracting values from the databases. This command allows the users to, at any time, see what variables have been determined by means other than user input, their values, and which rules (or databases) were used to determine them. Figures 12 through 14 show an example of this feature.

The user first selects which variables he/she wants to look at and then the system proceeds to show which rules determined them. Notice how it is possible for variables to be determined (or updated) by more than one rule. The user, of course, can choose any number of variables, though for this example only one variable, the cause of the anomaly, was selected. This feature not only gives the user complete control over the system, but allows him/her to see all the facts and knowledge that can be inferred from the inputs they have given, the available databases, and the expertise in the rules. As a final option, the user is also allowed, at any point, to exit from the program or begin a new session without ever leaving the program's window screen.

RESULTS

The diagnostic results are in the form of confidence factors derived from both the confidence assigned to rules by the experts and also the confidence of variables input by the user. Both the confidence in the rules/heuristics and the input of certainty factors by the user are needed to diagnose anomalies, as they contain vital knowledge which can only be represented as such. The results window is shown in Figure 15 (see next page).

The results window in our system includes, in addition to the cause(s) of the anomaly, the orbit of the satellite, whether input by the user or determined by rules, and a list of the causes considered in the diagnostics. The window can easily be modified to display any other

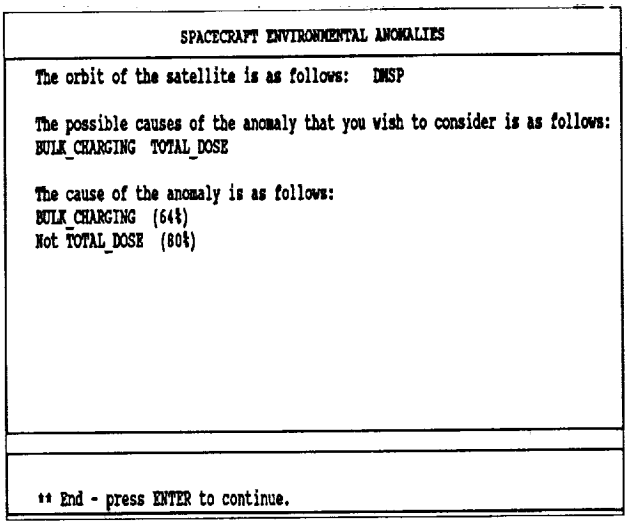


Figure 15. Results screen

information which is considered important. In the example, the cause of the anomaly was determined to be bulk charging with a confidence of 64%, and determined *not* to be total radiation dose with a confidence of 80%. The knowledge base does, of course, contain rules and formulae which can determine the cause of the anomaly with 100% confidence, or completely rule out a particular cause. For these situations the system will simply say that the cause, for example, *is* bulk charging or *is not* total dose.

The main concern with the system is the actual confidence and validity of the rules themselves. Since experts in any field are likely to disagree over certain areas, there may be rules to which other experts would apply slightly higher or lower degrees of confidence. This is certainly a consideration when using such a system, though it must be remembered that it is due to such a confidence/certainty question in the field that this type of expert system is needed. In general, as more quantitative environmental data become available in the immediate area of a spacecraft, we can apply the higher confidences to all of the system's rules. In addition, the features provided by the interface allow the user to see exactly what rules are being used so there is complete awareness and understanding of the formulae and knowledge being used.

An advantage of this particular system is that its interface is completely generic. Not only can the system run on many machines, the interface can be used in any field since the rules and knowledge base are completely independent of it. By substituting rules from another field, the system becomes an expert system for that field able to diagnose or solve problems towards which its tailored rules converge. In this sense the software is completely reusable.

FUTURE WORK

We are improving our EnviroNET network with the addition of an IBM RS/6000. Because the inference engine is machine independent and the remaining code is written almost entirely in C, the porting of the system to this UNIX machine will be quite simple. Once there, not only will the speed of the Expert System be increased, but with the use of X Windows, the system will also be enhanced.

For example, with X Windows the user could have one query window which prompts him/her for information, another separate window that displays which rules are being tested and fired, which variables are being searched for, and another window for graphics. With these multiple windows the user can see the entire system working at once and be freed from having to change windows to see system information.

CONCLUSION:

The EnviroNET expert system combines the algorithmic capabilities of mathematical programs and diagnostic models with expert heuristic knowledge, and uses confidence factors in variables and rules to calculate results with degrees of human confidence associated with them. Since the causes of environmentally induced spacecraft anomalies depend not only on algorithms, but also on environmental conditions, rules and information can rarely be known with 100% certainty. Based on present experiences, the role for the expert system is for either quasi-real time, or post analysis. There is a need to greatly improve our ability to predict the environment before meaningful work can be done in forecasting satellite anomalies.

ACKNOWLEDGMENTS

We are indebted to the staff of NASA's Johnson Space Center's AI Laboratory for their cooperation in the use of CLIPS. Funding was provided by NASA Headquarters and the Geophysical Laboratory (GL) Space Systems Environmental Interaction Technology Office and the Space Station Division of the U.S. Air Force under contract F04701-88-C-0089

REFERENCES

1. EnviroNET : An On-line Environmental Interactive Resource, Proc. Fourth Annual Space Operations and Research (SOAR) Symposium, June 26-28, 1990

2. 28th Aerospace Sciences Meeting, Reno, Nevada, January 8-11, 1990

A. Vampola, "Tutorial on Spacecraft Environmental Interactions Anomalies," CP AIAA 90-0172.

D.Wilkinson, "NOAA's Spacecraft Anomaly Data Base," CP AIAA 90-0173.

Robinson, "Anomalies Due to Single-Event Upsets," CP AIAA 90-0174.

D Gorney and H. Koons, "Spacecraft Anomaly Expert System," CP AIAA 90-0175.

J. Gaffey and D. Bilitza, "Trapped Radiation Model Facility," CP AIAA 90-0176.

H. Garrett and A. Whittlesey, "Environmentally Induced Spacecraft Anomalies on TDRSS," CP AIAA 90-0178.

3. Koons, H. C., and Gorney, D. J., "Spacecraft Environmental Anomalies Expert System: A Status Report," Aerospace Report # ATR-88 (9562)-1, The Aerospace Corporation, El Segundo, CA., 1 Dec. 1988.

4. Giarratano, J. C. (1989, May). "Artificial Intelligence Section," *CLIPS User's Guide, Version 4.3 of CLIPS.* Lyndon B. Johnson Space Center.

An Embedded Rule-Based Diagnostic Expert System in Ada.

Robert E. Jones
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Eugene M. Liberman
Sverdrup Technology Corporation
2001 Aerospace Parkway
Brook Park, Ohio 44142

ABSTRACT

Ada is becoming an increasingly popular programming language for large Government-funded software projects. Ada with its portability, transportability, and maintainability lends itself well to today's complex programming environment. In addition, expert systems have also assumed a growing role in providing human-like reasoning capability and expertise for computer systems.

This paper discusses the integration of expert system technology with Ada programming language, specifically a rule-based expert system using an ART-Ada (Automated Reasoning Tool for Ada) system shell. The Inference Corporation developed ART-Ada under a program sponsored by the NASA Johnson Space Center. The NASA Lewis Research Center was chosen as a beta test site for ART-Ada. The test was conducted by implementing the existing Autonomous Power EXpert System (APEX), a Lisp-based power expert system, in ART-Ada.

Three components, the rule-based expert system, a graphics user interface, and communications software make up SMART-Ada (Systems fault Management with ART-Ada). The rules were written in the ART-Ada development environment and converted to Ada source code. The graphics interface was developed with the Transportable Application Environment (TAE) Plus, which generates Ada source code to control graphics images. SMART-Ada communicates with a remote host to obtain either simulated or real data. The Ada source code generated with ART-Ada, TAE Plus, and communications code was incorporated into an Ada expert system that reads the data from a power distribution test bed, applies the rules to determine a fault, if one exists, and graphically displays it on the screen.

The main objective of this study, to conduct a beta test on the ART-Ada rule-based expert system shell, was achieved. The system is operational. New Ada tools will assist in future successful projects. ART-Ada is one such tool and is a viable alternative to the straight Ada code when an application requires a rule-based or knowledge-based approach.

Keywords: Ada, Expert System, Rule-Based, Knowledge-Based.

INTRODUCTION

Ada is becoming the language of choice for large Government-funded projects as increasing software size and complexity drives the cost of development and maintenance upward. Ada (J.G.P Barnes, 1984) language offers standardization, portability, maintainability, and readability. All these features provide an excellent environment for developing complex software, increasing programmers' productivity, and encouraging team work.

Many large-scale software projects targeted toward space applications involve health monitoring and control of mission-critical systems. The necessary expertise to maintain reliable space mission operations is often unavailable due to limited resources. Even if the expertise is available, routine system health monitoring remains a time-consuming and tedious task. Expert systems have been taking an increasingly larger role in providing knowledge for problem resolution and in handling tedious tasks.

The integration of expert system technology with Ada programming language results in a powerful combination for use in many large-scale computer applications. An expert system shell that generates Ada code is one such combination. ART-Ada is an expert system shell that generates Ada code.

This paper describes the techniques for and implementation of a rule-based expert system using the ART-Ada expert system shell. The Inference Corporation developed ART-Ada under a program sponsored by the NASA Johnson Space Center. The NASA Lewis Research Center was chosen as a beta test site for ART-Ada. The test was conducted by implementing the existing Autonomous Power EXpert system (APEX) (Ringer and Quinn, 1990) with ART-Ada. APEX is a Lisp-based power expert system designed as a fault diagnostic adviser for the monitoring and control of 20 kHz power distribution test bed. APEX is being developed at the NASA Lewis Research Center.

SYSTEM DESCRIPTION

The system description is limited to a discussion of the software implementation of SMART-Ada and the hardware on which the software was implemented. SMART-Ada is written on a SUN SPARCstation 1 equipped with 16 megabytes of memory and approximately 0.5 gigabyte of hard disk capacity.

The operating system is SUN OS 4.0.3c running X Window X11R3. The following software packages were used to implement SMART-Ada:

(1) ART-Ada
(2) Transportable Application Environment (TAE) Plus
(3) Remote procedure call protocol (RPC)
(4) X Window system
(5) X Window Ada bindings
(6) VERDIX Ada compiler

ART-Ada is an expert system shell for the development of rule-based or knowledge-based expert systems. ART-Ada is based on the Automated Reasoning Tool for Information Management (ART-IM). One important feature of ART-Ada is its ability to generate Ada source code from a knowledge-based or rule-based application written in ART-IM. The syntax of ART-Ada is compatible with ART-IM, making code developed in ART-IM transportable to ART-Ada as long as no machine-dependent features of either software are used.

TAE Plus is a software package developed by the NASA Goddard Space Flight Center. The package is an integrated environment for creating and running window-based applications with a graphical point-and-click user interface. TAE Plus is based on the X Window System. TAE Plus was chosen for the SMART-Ada graphical user interface because it can generate Ada code.

RPC is a communications protocol developed by Sun Microsystems. This protocol allows machines of different types to interact with each other on a procedure level. This interaction means that one machine can call a procedure on the other, pass arguments to the procedure and then receive any returned values. RPC software for SPARCstation 1 was developed in the C programming language. An Ada-to-C interface is required to allow SMART-Ada software to take full

advantage of RPC protocol. This interface was written for this project. The X Window System, developed at MIT, has become the industry standard and runs on a wide range of computing and graphics machines. The X Window System provides a powerful windowing environment for producing high-performance graphical interface. Since the X Window System was developed in C programming language, an interface between the Ada code and the X Window C code is necessary to take advantage of the X Window System features.

Ada language bindings to the X Window System are a package developed by Science Applications International Corporation (SAIC). The bindings provide the necessary interface between X Window C libraries and the Ada code in SMART-Ada. The use of bindings allows the Ada code to take full advantage of the powerful X Window System procedures.

The VERDIX compiler is used for compilation and executable code generation. The VERDIX compiler was recommended by the Inference Corporation for development and implementation of SMART-Ada. The VERDIX compiler also allows for a good Ada-to-C interface, which is an important feature needed for accessing X Window System procedures with the Ada code.

SMART-Ada system components

The SMART-Ada system consists of three major components:

(1) A rule-based expert system
(2) A graphics user interface (GUI)
(3) Communications software

These three components were integrated to make up the entire SMART-Ada system. All components are implemented in Ada programming language. However, in two cases the interface between C libraries and the Ada code is used to enable Ada to access existing C software libraries. The interface between X Window procedures and the Ada graphics programs was implemented with the SAIC Ada language bindings to the X Window System.

The interface between RPC software and Ada was a part of SMART-Ada development. Each component and its implementation is discussed in greater detail in the following paragraphs.

Rule-Based expert system

The knowledge-based rule set was developed with ART-IM because of its user-friendly development environment. ART-IM's powerful debugging features were heavily utilized.

The rule-based expert system is responsible for monitoring the operational state of a 20 kHz power distribution test bed. If an abnormality occurs in the test bed, the expert system detects the fault condition and isolates the probable cause. It performs fault detection by comparing measured operating values to the expected values, accessing information and rules contained within its knowledge base in order to isolate the fault.

A collection of rules in the knowledge base forms groups of logical subtasks. The logical subtasks are connectivity, initialization, detection, isolation, affected loads, and recommended actions. The connectivity subtask determines the power distribution configuration of the 20 kHz power distribution test bed. The initialization subtask calculates expected values for voltages, currents, and power on the basis of the test bed configuration and the physical properties of the test bed components. The detection subtask compares the expected values obtained in the initialization subtask with the actual system values obtained from the test bed. The isolation subtask isolates the problem and assists the expert system in determining where in the circuit problems have occurred. The affected loads subtask determines which load is affected by the faults in the system. The recommended action subtask, in the future development of the system, will recommend a new

configuration for the power distribution or will automatically reconfigure the power distribution to alleviate existing problems.

The order of rule execution in a subtask is determined by the salience of each rule. The higher salience rules execute first and the lower salience rules execute last. The last rule in the subtask contains the lowest salience so that it will execute only after all the rules in the subtask have executed. In SMART-Ada, the lowest salience rule in each subtask is used to "clean up" the current subtask and set the environment for the next subtask. Since SMART-Ada is a monitoring system, the rule-based expert system must execute continuously. The last subtask sets the environment for the first subtask, creating the effect of an infinite loop.

Expert system and Ada Interface

In order to execute Ada language subprograms from ART-Ada, an interface between ART-Ada and Ada language is required. The interface is accomplished by defining an Ada USER package (Figure 1). Within ART-Ada, USER is a reserved symbol for accessing external Ada code from ART-Ada rules. Parameter passing is also possible between ART-Ada and subprograms in the USER package. The parameters, however, have to adhere to the syntax of ART-Ada. SMART-Ada uses subprograms contained in the USER package to control the graphical user interface and to obtain data from the 20 kHz power distribution test bed.

ART-Ada contains a feature that allows one function to be specified as asynchronous. A function defined as asynchronous is automatically invoked before and after each rule execution. The subprogram named ASYNCH_FUN in the USER package is defined as an asynchronous function. ASYNCH_FUN is responsible for reading data from the test bed or using simulated data and providing data to the rule-based expert system (Figure 2) as well as to the graphics interface. All subprograms in the USER package are capable of accessing and modifying data in the ART-Ada code.

As the rules are being executed, the asynchronous function waits until the last rule has been completed. The last rule execution is an indicator for ASYNCH_FUN to read new data from the 20 kHz power distribution test bed and introduce the new data to the expert system for evaluation. The new data are also dispatched to the graphics interface for a system status update. The execution of the expert system then continues with execution of the first subtask.

The WRITE_TO_FILE subprogram in the USER package is responsible for writing the explanations of errors that occur as a result of the rule-based expert system execution. The rule-based expert system provides the file name and the line-by-line text that is to be entered in the file specified in the file name parameter. The file name is determined by which rule subtask is executing at the moment. The detection rule subtask, for example, writes to the DETECT.DAT file, and the isolation rule subtask writes to the ISOLATE.DAT file. The text that goes in a file is determined by the executing rule. The FLAG_ERROR subprogram is invoked when an error condition is present. The rule that invokes the subprogram must provide the switch name and the faulty component to the subprogram. The code in the FLAG_ERROR subprogram communicates by means of the graphics interface and sets the visual alert in the graphics module.

Graphics User Interface

The graphics user interface (GUI) for the SMART-Ada was developed with TAE Plus. TAE Plus was chosen for this application because of its capability to generate Ada code for the developed graphics.

The DDO (data driven object) feature of TAE Plus is used to achieve the dynamic display of the system status. The screens and the DDO's were created with the TAE Plus

graphics editor. The screens show the system status at high and low levels. The high- level screen shows the overall system state and configuration. The faults with the system components are shown with appropriate color and a warning banner. The lower-level screens provide a more detailed look of a selected component. The currents, voltages, and power indicators contain the up-to-date values. The faults with any of the values are also marked with appropriate colors and a warning banner.

An explanation for the errors is also provided. The Rule-Based expert system generates the text corresponding to the system status. The text is stored in the file set up for that purpose. The file DETECT.DAT contains the result of the detection subtask execution, the file ISOLATE.DAT contains the results of the isolation subtask execution, etc. The GUI displays the contents of these files to provide the user with the explanation of the system status.

Communications Software

SMART-Ada communicates with a remote host to obtain either simulated or real data. The communications link between SMART-Ada and the data acquisition software on the remote host is accomplished by using the transmission control protocol / internet protocol (TCP/IP) and RPC protocol. The remote host must contain the procedure that is invoked to provide data to SMART-Ada. The remote procedure returns system status data that are interpreted by the system.

BETA TEST RESULTS

The main objective of SMART-Ada implementation was to conduct a beta test on the ART-Ada rule-based expert system shell. The beta test revealed that it is, indeed, possible to implement an Ada-based application by using the ART-Ada expert system shell. However, a few problems were found as the project developed. The first problem occurred in an attempt to deploy a set of rules. The set of rules was executing properly in the development environment, but when the rules were converted to Ada source code and an executable code was generated, the program failed to run. A constraint Ada error was raised when execution was attempted. The problem was reported to the Inference Corporation who acknowledged and fixed the problem and sent us corrected version of ART-Ada. The second problem was found with the ART-Ada package. A variable contained no value after a value had been assigned to it. The Inference Corporation has acknowledged the problem and promised to fix the package for the next version of the software and has suggested a way to work around the problem.

The SMART-Ada system which was developed consisted of approximately 70 rules in ART-Ada. The code for manipulation of the switches generated approximately 340 Ada statements. The interface between ART-Ada and Ada consisted of approximately 200 Ada statements, while the Ada Graphics interface produced about 2100 statements. The completely deployed system including system overhead produced 13,484 Ada statements total. Further optimization of the ART-Ada code could reduce this number slightly.

The maintainability issues of the ART-Ada code must be addressed. Unfortunately, the tendency is to address maintainability of the Ada code generated from the ART-Ada application. The proper way to maintain the ART-Ada application is through the ART-Ada code itself. System maintenance will become much easier if emphasis is placed on ART-Ada code maintenance rather than on Ada code maintenance.

CONCLUDING REMARKS

As Ada programming language becomes more and more prevalent, new Ada tools will assist in making Ada projects even more successful. ART-Ada is the first rule-based tool available for Ada programming language. The ART-Ada package has its problems, like any initial version of a major software

release. But the authors feel that ART-Ada is a viable alternative to the straight Ada code for an application that requires a rule-based or knowledge-based approach.

ACKNOWLEDGMENTS

The authors would like to thank Todd Quinn of Sverdrup Technology Corporation for his invaluable technical and editorial assistance and the entire Space Electronics Division AI laboratory staff for their comments and support.

REFERENCES

1. Barnes, J.G.P (1984). Programming in Ada. 2nd ed., Reading, MA, Addison-Wesley.

2. Ringer, M.J. and Quinn, T.M. (1990), Autonomous Power Expert System. NASA CR-185263.

```
with ART;
package USER is

        function ASYNCH_FUN;

        procedureWRITE_TO_FILE
        (NAME_OF_FILE :in ART.ART_OBJECT;
                STRING_VALUE:in ART.ART_OBJECT);

        procedure FLAG_ERROR
                (SWITCH_NAME:in ART.ART_OBJECT;
                 COMPONENT:in ART.ART_OBJECT);

        end USER;
```

Figure 1. Ada USER Package

```
with USER; use USER;
(def-user-fun asynch-fun
        :args ()
        :returns                :void
        :compiler               :verdix)

(def-user-fun write-to-file
        :args (
        (FILE-NAME    :ART-OBJECT)
        (STRING_VALUE       :ART-OBJECT))
        :returns                :void
        :compiler               :verdix)

(def-user-fun flag-error
        :args (
        (SWITCH       :ART-OBJECT)
        (COMPONENT :ART-OBJECT))
        :returns                :void
        :compiler               :verdix)

        .
        .
        .
        (set-asynch-func async-func)
```

Figure 2. The rule-based expert system use of USER package

N93-11929

INTEGRATION OF TASK LEVEL PLANNING AND DIAGNOSIS FOR AN INTELLIGENT ROBOT

Amy W. Chan
UFA, Inc.
335 Boylston street,
Newton, MA 02159

Work performed under Contract# NSA8-38420

Abstract

This paper is to diagnose a satellite floating through space with a telerobot attached performing maintenance or replacement tasks. This research included three objectives. The first objective was to generate intelligent path planning for a robot to move around a satellite. The second objective was to diagnose possible faulty scenarios in the satellite. The third objective included two tasks. The first task was to combine intelligent path planning with diagnosis. The second task was to build an interface between the combined intelligent system with Robosim.

The ability of a robot to deal with unexpected scenarios is particularly important in space since the situation could be different from time to time so that the telerobot must be capable of detecting that the situation has changed and the necessity may exist to alter its behavior based on the new situation.

The feature of allowing human-in-the-loop is also very important in space. In some extreme cases, the situation is beyond the capability of a robot so our research project allows the human to override the decision of a robot.

Introduction

This research project is the Phase II SBIR project, "Integration of Task Level Planning and Diagnosis for an Intelligent Robot". This research project is an intelligent system developed for multiple processes running in real time. The intelligent system includes three parts: AI Path Planner, Diagnosis and the interface with Robosim *1. Refer to Figure 1-1 for the relationship.

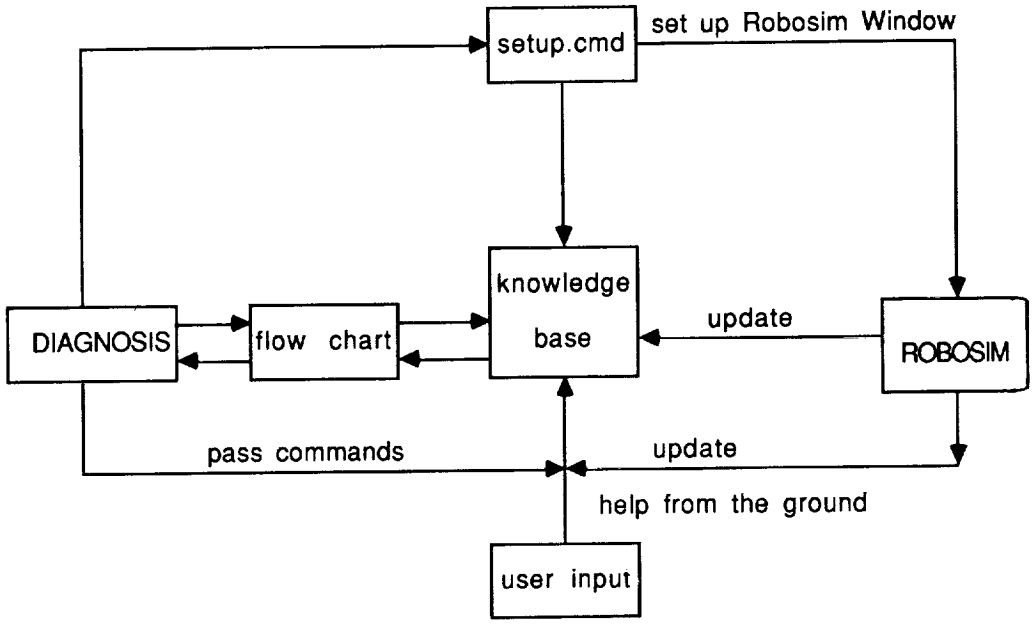


Figure 1-1

In addition to Robosim window, there are two windows in the system: the Satellite Window and the Knowledge Window. A simplified generic satellite model floating through space with a telerobot attached is simulated in the Satellite Window.

There are two knowledge bases in the system: the Satellite Knowledge Base and the Diagnosis Knowledge Base. Both are constantly updated during the simulation and displayed in the Knowledge Window. The Knowledge Window also displays simplified physical architecture and logical diagnosis of satellite subsystems.

The intelligent system provides user friendly environment. It is written in C and runs on a Silicon Graphics Iris workstation with Unix operating system.

AI Path Planner

The AI Path Planner includes three parts: a scenario definition file - "model.dat" and scenario parser; model builder; and Path Planner. Refer to Figure 1-2 for the relationship.

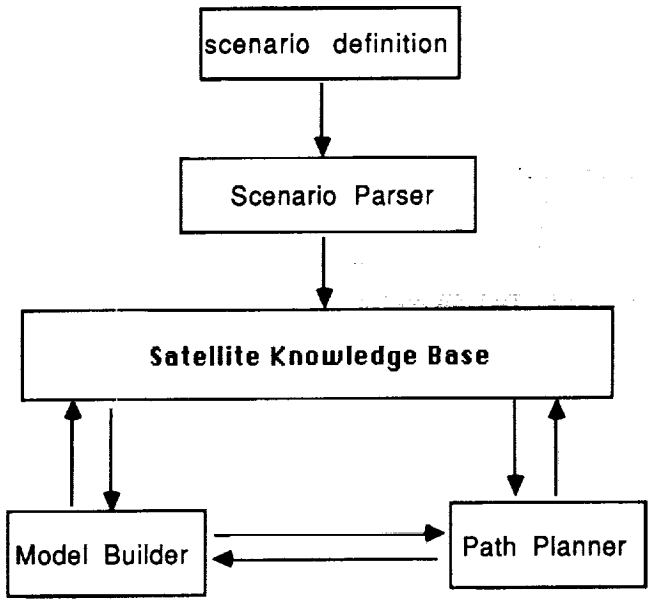


Figure 1-2

Scenario Definition File and Scenario Parser

A scenario definition file called "model.dat" was created to define the geometric configuration of a simplified generic satellite model. The scenario file defines the brief title of each satellite module and assigns each subsystem, panel, antenna and the robot to a separate module. The scenario file also assigns a list of parts to some modules and defines all the possible moves of each module.

There are two parsers in the system: the Scenario Parser and Setup Parser. The Scenario Parser was created to read and store the scenario definition file to the Satellite Knowledge Base. The Setup Parser was created to read a setup file for Robosim.

The Scenario Parser also calculates the center of each module, calculates the distances of each module to all its possible moves, sets up the constraint condition and stores them into the Satellite Knowledge Base.

Model Builder

The model of a simplified generic satellite was constructed in the form of three cylinder-like structures piled up together. Each cylinder-like structure has six sides. The total modules of the satellite model is the eighteen sides plus the top and bottom of the model. The Model Builder displays the model with a pair of solar panels, an antenna and the robot to the Satellite Window.

Path Planner

The Path Planner determines the best path for the robot to move from the module which the robot is standing at to any other target modules. Once a move command is given to the robot, the Path Planner will search dynamically all the adjacent paths one by one and generate the best candidate path set. Each path set includes one or more module elements. The determination of each candidate path set is a complicated task that involves calculating the total distance of the path set, avoiding obstruction, paying extra costs and penalties.

The search algorithm is First Depth plus heuristic and optimal. To be more specific, in order to get the shortest move, when a target module is given, the Path Planner calculates and compares all the distances from the current module to all the adjacent modules one at the time and get the shortest one.

Once the shortest move is found, the Path Planner pushes the target module and its total distance into a result stack. At each dead end, the Path Planner pops the last node off the stack and tries a new path from that point (backtracking).

After the candidate path is determined, the Path Planner pops up the stack element by the rule of "first-in-last-out" and displays them into the Satellite Knowledge Base Display in the Knowledge Window. To simulate the intelligence of the system, the following features had been added to the Path Planner.

- If the robot moves from or to the top or bottom module, a certain "cost" will be charged. For example, if the robot is located at the top module - 'S' and going to move to any adjacent modules, or the robot is standing at any adjacent modules and moving to the top module - 'S', the Path Planner adds an extra cost to the total distance.

- There are a list of parts assigned to some modules. After a move path set is generated, the robot moves one by one according to the path set. While the robot moves to each module element, the Path Planner checks whether a part is assigned to the current module. If there is a part assigned to the current module, the Path Planner will check whether the robot is carrying that part. If the robot is carrying that part, it will skip that part. If the robot is not carrying that part, it will pick up that part.

- If the robot passes some modules with constraints in the center such as panels or antenna, it will be fined with a certain penalty. For example, if the robot passes a module with the panel in the center, the Path Planner will add an extra 15 second penalty to the total penalty. If the robot passes a module with the antenna in the center, the Path Planner will add an extra 20 second penalty to the total penalty.

Unexpected Events

To simulate an unexpected event, an obstruction storm was created. The obstruction storm includes three obstructions. Each obstruction has different sizes. The three obstruction were flying in three different directions in three different speeds in the simulation. The system can handle the following unexpected events:

- An obstruction can be selected to fall on the surface of the model. If the obstruction is falling on the module at which the robot is standing, the robot will detect it and move away to avoid being hit.

- If a given target for the robot to move has an obstruction, the system will detect it , discard the move command and pop up an error message to the Satellite Knowledge Base in the Knowledge Window.

- The Path Planner will detect the module with an obstruction and skip it from the path planning.

Human-in-the-loop

The system allows the human being to take over the control of the robot during the simulation. The system allows the human being to input his path set and also allows the human being to review or change the path set. After the path set is determined, the robot moves one by one according to the path set. The following error checking were performed:

- If one module element of the human input path set is not alphabetic, the system discards it from the path. The new path set remains valid.

- If one module element of the human input path set cannot be found in the Satellite Knowledge Base, the system discards the whole path set and displays an error message to the Satellite Knowledge Base Display in the Knowledge Window.

- If one module element of the human input path set has an obstruction, the system discards the whole path set and displays an error message to the Satellite Knowledge Base Display in the Knowledge Window.

Diagnosis + Path Planner + Robosim

We have combined the AI Path Planner with the Diagnosis and built the interface between the new intelligent system with the Robosim. We pass the Robosim commands through the Unix feature - pipe to the Robosim.

To setup the Robosim, users have to define the color and view point in a file called *"setup.cmd"*. Users can also create, translate or rotate objects in this file. The Robosim initializes the robosim screen according this file.

The Diagnosis includes space experiment lab and four subsystem diagnosis. There is an option in the main menu for each subsystem. Once a particular subsystem is selected, the Diagnosis checks the Satellite Knowledge Base and finds out which module is assigned. Once the module is found, the Diagnosis calls the Path Planner to generate the best path and the robot will move one by one according to the path set. The Satellite Knowledge Base is updated simultaneously.

At the same time, a simplified physical architecture chart of that subsystem is displayed on the left side of the Knowledge Base Window and the complete diagnosis flow is displayed on the right side of the window.

The Diagnosis passes the predefined command files to Robosim to set up the Robosim window simultaneously. Each part of the physical architecture is drawn as a rectangle. As the Robosim does not allow displaying words, the labelling of the Robosim is displayed on the left Knowledge Window.

To cover all the failure scenarios, a random number is generated to give a positive or negative answer at each determination point during the diagnosis of each subsystem. After a complete failure scenario is diagnosed, the system displays it on the right Knowledge Window.

Once a faulty part is detected during the diagnosis, the Diagnosis sends a command file to Robosim to command the robot in the Robosim Window to identify the faulty part. If there are more than one faulty parts found, the robot will identify them one by one. In

some cases, the Diagnosis is not able to find the faulty part, the robot asks for help by displaying a big "HELP" in the Robosim Window.

Conclusions

This scientific AI project includes multiple processes and each process is displayed in a separate window. This artificial program can integrate with a simulator like Robosim.

The breadth-first search plus heuristic and backtracking strategy enables this project to provide a shortest path set between any two modules in the satellite.

This project handles unexpected events and allows human to override the decision of an intelligent robot during the simulation.

Acknowledgments

I appreciate all of the help that made this project possible and a success. Many thanks go to Ms. Cynthia A. Coker, the C.O.R of this research project. I would like to thank Kiet Huang, our previous project manager, for his accomplishments during Phase I and his many inputs and ideas at the beginning of Phase II.

I also appreciate Csaba A. Biegl Ph.D. from Vanderbilt University for his valuable explanation of the ROBOSIM and the integration of our system with Robosim. Finally, I would like to thank Dr. Harold L. Alexander from M.I.T. for his valuable consultation of the satellite subsystem diagnosis.

Reference

1. The initial Robosim was created by Ken Fernandez, Ph.D. of MSFC, NASA. The Robosim which this research project integrated with is an intelligent agent of the Robosim developed by Csaba A. Biegl, Ph. D. from Vanderbilt University.

N93-11930

Intelligent Fault Management for the Space Station Active Thermal Control System

Tim Hill
McDonnell Douglas Space Systems Company
Space Station Division
16055 Space Center Boulevard
Houston, TX 77062
TIMHILL@NASAMAIL.NASA.GOV

Robert M. Faltisco
McDonnell Douglas Space Systems Company
Space Station Division
16055 Space Center Boulevard
Houston, TX 77062
RFALTISCO@NASAMAIL.NASA.GOV

Abstract

This paper describes the Thermal Advanced Automation Project (TAAP) approach and architecture for automating the Space Station Freedom (SSF) Active Thermal Control System (ATCS). The baseline functionality and advanced automation techniques for Fault Detection, Isolation, and Recovery (FDIR) will be compared and contrasted. Advanced automation techniques such as rule-based systems and model-based reasoning should be utilized to efficiently control, monitor, and diagnose this extremely complex physical system. TAAP is developing advanced FDIR software for use on the SSF thermal control system. The goal of TAAP is to join Knowledge-Based Systems (KBS) technology, using a combination of rules and model-based reasoning, with conventional monitoring and control software in order to maximize autonomy of the ATCS. TAAP's predecessor was NASA's Thermal Expert System (TEXSYS) project which was the first large real-time expert system to use both extensive rules and model-based reasoning to control and perform FDIR on a large, complex physical system. TEXSYS showed that a method is needed for safely and inexpensively testing all possible faults of the ATCS, particularly those potentially damaging to the hardware, in order to develop a fully capable FDIR system. TAAP therefore includes the development of a high-fidelity simulation of the thermal control system. The simulation provides realistic, dynamic ATCS behavior and fault insertion capability for software testing without hardware related risks or expense. In addition, thermal engineers will gain greater confidence in the KBS FDIR software than was possible prior to this kind of simulation testing. The TAAP KBS will initially be a ground-based extension of the baseline ATCS monitoring and control software and could be migrated on-board as additional computational resources are made available.

Introduction

Thermal control systems are very complicated to operate in both nominal and off-nominal states. This problem is compounded in space. The Space Station Freedom (SSF) Thermal Test Bed (TTB) at the National Aeronautics and Space Administration's (NASA) Johnson Space Center (JSC) in Houston, Texas is one such complex system. The operation takes several thermal experts (typically thermal engineers) as well as a staff of technicians, safety officials and other support personnel. In short, the TTB is expensive to operate and technically complex, particularly in off-nominal states.

For this reason and due to the lack of plausibility of having such a TTB support staff on the space station, the need for autonomous operation of the Space Station Freedom Project's (SSFP) Thermal Control System (TCS) is apparent. However complicated the task of autonomous monitoring and control of the SSFP TCS, it is necessary to have a high degree of automation due to the harsh environment of space and the time which astronauts would spend on such "housekeeping" functions (e.g. monitoring temperatures, opening valves, etc...) without such system autonomy.

It is this problem that NASA began addressing in the System's Autonomy Demonstration Program's (SADP) Thermal Expert System (TEXSYS) project. The TEXSYS project was developed in the middle to late 1980's and was demonstrated in August of 1989. The project's major accomplishment was that it was the first real-time expert system to autonomously operate and diagnose faults in a system as complex as the thermal testbed TCS.

TEXSYS consisted of several software/hardware modules which enabled the expert system to accomplish its task. Figure 1 shows the architecture of the TEXSYS project. The overall job of TEXSYS was accomplished via the use of 4 separate computers. The expert system was on a Symbolics 3650 (denoted TEXSYS SYMBOLICS) and was the "brains" of operating the TTB.

The TEXSYS software diagnosed 17 faults and operated the thermal testbed ATCS accomplishing system startup, shutdown and temperature set-point change. The expert system utilized model-based reasoning with qualitative modeling and hypothetical reasoning via an assumption-based truth maintenance system [Glass 90]. Rules were used in conjunction with the models to accomplish diagnosis of most of the faults successfully isolated. The expert system was relatively large in terms of memory usage (on the order of several megabytes of memory).

The TEXSYS prototype diagnosed faults and controlled the behavior of a GROUND-based system in the TTB. It is desirable to utilize this technology in a space-based system located on the Space Station itself. The constraints of resources (memory and power) on the Space Station will not allow for such a large software system to be implemented. In addition, a high cost was incurred due to the use of such a complex hardware system for debugging and testing. The notion of using a simulation of the ATCS was not fully explored in TEXSYS.

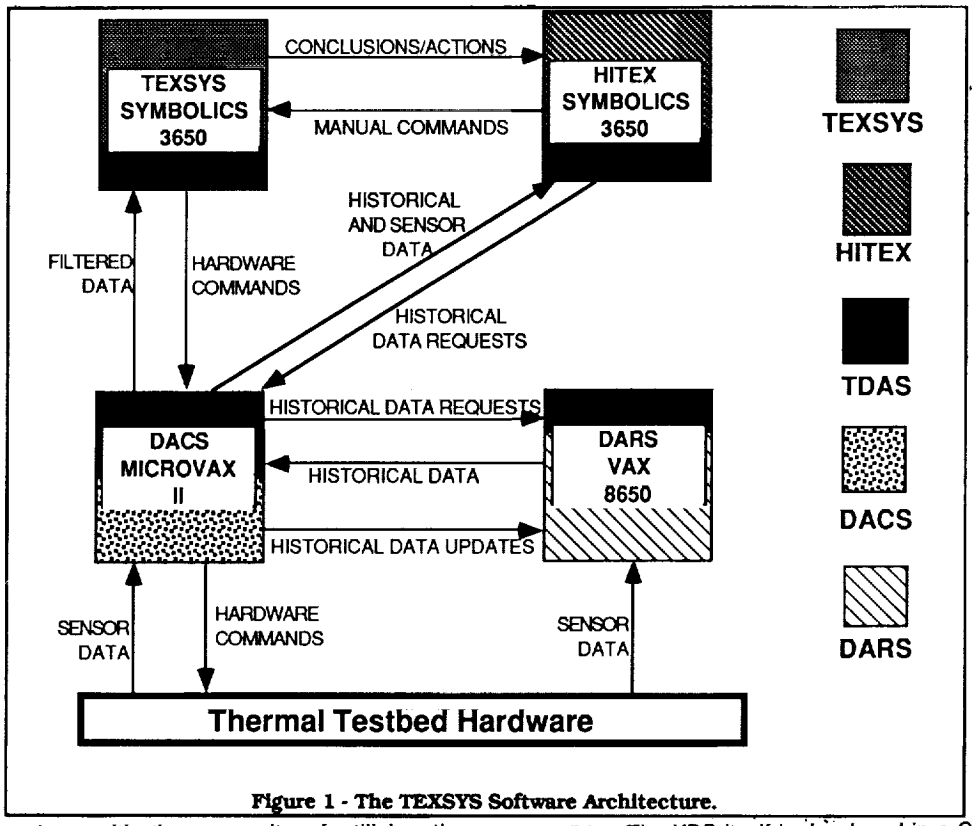


Figure 1 - The TEXSYS Software Architecture.

These reasons along with the necessity of utilizing the technology of TEXSYS for SSFP ATCS operation made follow-on work to TEXSYS important if not inevitable. NASA Headquarters' Advanced Development for SSFP initiated the Thermal Advanced Automation Project (TAAP) to address these problems.

The goals of TAAP include the development of a knowledge-based system (KBS) to cover all known faults on the Thermal Control System, a greater number of faults than TEXSYS covered. Also, to detect inconsistent behavior which cannot be attributed to a known fault. A high-fidelity simulation of the thermal control system will be developed and utilized for testing of the KBS. The architecture and the hardware and software platforms of the SSFP's Data Management System (DMS) should be adhered to whenever possible. In addition, since the contracting organization responsible for developing the SSFP Thermal Control System is doing the TAAP work, communication with the SSF team is greatly increased (i.e. McDonnell Douglas Space Systems Company - Space Station Division is building the TAAP systems).

The TAAP project addresses these issues. A high-fidelity simulation is under development with particular attention paid to the simulation of faults which are used for the testing of the Fault Detection, Isolation, and Recovery (FDIR) KBS. The simulation is being developed with a code called the Advanced Thermal Hydraulic Energy Network Analyzer (Athena) [ATHENA 86] which performs thermal-hydraulic calculations required to simulate operation of the Active Thermal Control System (ATCS). The plans are to run the simulation in a batch/file mode with real-time (or near real-time) capability added later.

TAAP uses a similar architecture and platform to the Space Station's Data Management System. That is, an Intel 80386 based computer with Ada-based code is utilized when possible. The KBS itself is developed in a C-based language called CLIPS (C Language Integrated Production System). CLIPS has been ported to provide to an Ada-based version. The simulation and KBS portions will be connected via a Space Station-like database system. This system is also developed in Ada.

More details on the TAAP approach are given in the ATCS Advanced Automation section. The purpose of this introduction was to provide an understanding of the origins of the concept of automating the Space Station Thermal Control System and an introduction to the TEXSYS and TAAP approaches.

Space Station Freedom Thermal Control System Overview

The Thermal Control System (TCS) for Space Station Freedom (SSF) will provide the heating and cooling control necessary to maintain elements, systems, and components within their required temperature ranges. The SSF TCS is comprised of systems for passive and active thermal control. Passive thermal control is performed through the use of coatings, insulation, isolators, and selective placement of heaters. Active thermal control includes both internal and external systems. The internal thermal control system collects and transports waste heat to the external system from the habitation modules, resource nodes, and airlocks. The external active thermal control system consists of a thermal bus, its control hardware and software, and a set of space radiators. The external system is the focus of the TAAP, and is referred to here as the ATCS.

The ATCS is a central facility, transporting waste heat away from crew quarters, experiment packages, computers, etc., and radiating it into space. It utilizes ammonia as the working fluid and interfaces via heat acquisition devices

(HADs) with the habitation and laboratory modules, and pallet mounted equipment where the heat dissipation rates are too high to be controlled passively. HADs are heat exchangers that remove heat from fluid systems and electronic equipment directly. Liquid ammonia is supplied to the HADs by the ATCS and is vaporized by the particular heat load being serviced. The vapor is transported to the radiators which reject heat to space. Figure 2 shows a configuration of the major ATCS components on SSF.

Key pumping and control elements are the Rotary Fluid Management Device (RFMD) and the Back Pressure Regulating Valve (BPRV). The RFMD is a special centrifugal pumping device designed to separate liquid from vapor and operate in zero-gravity. Temperature control of the system is accomplished by control of RFMD drum pressure. The BPRV regulates saturation conditions in the main chamber of the RFMD by regulating pressure over a range corresponding to the desirable temperature set point of the system. Both of these components exhibit highly nonlinear nominal behavior which makes constructing dynamic numerical simulations difficult.

The FDIR Automation Problem

Performing Fault Detection, Isolation, and Recovery means monitoring and identifying fault conditions through interpretation of sensor inputs and calculated data. System control is required for executing the approriate actions to recover from the condition and return to a nominal operating mode, if possible. In an automated ATCS system, monitoring and control (M&C) and FDIR functions will be integrated. The system will be able to detect and diagnose the causes of faults, and then request confirmation or automatically invoke appropriate recovery or reconfiguration procedures.

The need for automation is clear. Complex space-based systems require constant monitoring of health information. Human operators scan telemetry for slight deviations from expected norms. In real-time, continuous operations this is very expensive. For shuttle operations, many systems require several engineers to monitor parameters, configuration, and component health changes for anomalies. These activities continue 24 hours a day during a mission. Shuttle missions last only a few days, while the SSF is projected to be in orbit for 30 years. By automating some or most of the FDIR functions the need for constant direct human monitoring and intervention will be decreased.

It is difficult to overstate the complexity of monitoring and diagnosing continuous variable dynamic systems in which few system parameters are observable. A variation of the General Diagnostic Engine (GDE), a model-based reasoning algorithm, was applied to a prototype space station thermal bus on the TEXSYS project. It was found that maintaining multiple fault hypotheses in currently available truth maintenance systems made real-time (approx. 1 minute) performance very difficult to achieve. Fault detection capability is also dependent on sensor distribution and variety. Less instrumentation increases the complexity of obtaining data for FDIR and increases reliance on algorithms to infer the state of a component from less direct sensor information.

Baseline Approach

This section describes the current issues facing the automation of the SSFP ATCS. The SSF team has recently undergone a Congressionally mandated restructuring effort. This effort has changed the particulars of the SSFP ATCS design including the design of the monitoring and control software which is responsible for the FDIR. However, the general choice of software techniques seems to remain the same, so this technique is described. Additionally, the constraints which led to this design concept are described and the relative advantages and disadvantages of such an approach are given.

This redesign affects software in a very large way. The number of onboard processors is now reduced to two Intel 80386 machines for initial phases of the SSFP. The bottom line for the monitor and control (M&C) software of the TCS is that there will be approximately 80kBytes of memory for the entire system. This poses a major constraint. It was decided that the M&C software would be divided into two parts, an onboard portion and a ground-based portion. The

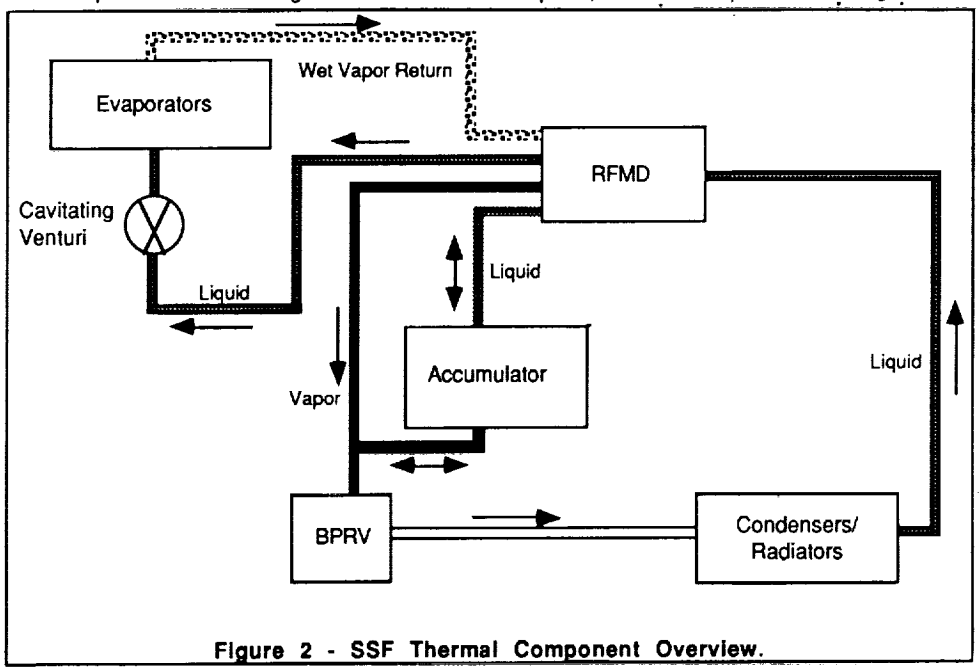


Figure 2 - SSF Thermal Component Overview.

onboard portion would have the task of handling the safety-critical faults, faults which must be isolated and/or recovered from immediately due to the severity of the fault's consequences. The portion moved to the ground would handle all other faults which are not as constrained by criticality or response time.

The baseline approach, prior to the restructuring, has been a table-driven design and it is likely this same technique will continue as the initial design for the M&C software. Basically, an event (e.g. sensor out of limits), triggered by the DMS will trigger the FDIR. The FDIR software will execute, using table data applicable to the event. This technique places the fault signatures across the rows of the table. An executive module matches the sensor data to one of these rows which contain sensor data approximations of what the fault corresponding to that row looks like (the fault signature). Each column will correspond to a particular sensor or other data item. When a match is made, that row's fault is isolated and the pertinent recovery action is taken.

This sort of table driven or table lookup approach fits well with the constraints when limited to safety critical faults only. It takes up small amounts of memory and storage, is relatively fast, and can isolate all *enumerated* faults. There are some problems with this approach that need to be examined.

One problem is that this can only diagnose faults written into the system. Multiple faults are not typically enumerated and if a sensor fails, the fault signatures change for all faults using that particular sensor, or, if the sensor reads bad information, it could diagnose a fault that isn't there. The table-lookup mechanism cannot, by itself, handle a problem not explicitly coded in the table.

It's important to have a mechanism for sensor failures to be contained. Sensor noise could force a false positive of a particular fault signature. Without a sensor check mechanism, a false recovery action could be taken. If a sensor fault is allowed to persist, which is quite likely due to the lack of reliability of sensors, particularly thermocouples, the ability to isolate even the enumerated faults is greatly diminished, if not eliminated.

Another major concern is the problem of FDIR on systems which have been reconfigured. Failed sensors and components could take months to replace. The tables would need to be coded for every newly reconfigured system state. This can lead to a combinatorial explosion of tables. Each fault multiplied by each sensor, multiplied by each new system state is quite a large number of tables. Considering the difficulty of enumerating all the faults, or all the system reconfigured states, this is quite a large problem.

However problematic the baseline approach is for handling FDIR on the SSF ATCS, it is well thought out considering the computational restraints onboard. With regards to onboard design and emergency fault handling capability, there may initially be no other choice. The major issue is that a predominant amount of faults will be handled in ground-based systems. This ground based portion of the system must avoid most of the problems with table lookup enumerated above. The automation of FDIR requires the use of the most advanced programming techniques available.

ATCS Advanced Automation

Advanced automation includes techniques and applications for monitoring, control, and FDIR. Some of the criteria used for determining the degree of automation are:

- safety,
- lowered operations life cycle costs,
- development cost,
- efficiency (ground controller and crew productivity),
- reliability,
- technology maturity,
- integration with existing controls,
- operational support requirements,
- human factors concerns (response time, task monotony),
- impact to established schedules and baselines,
- chances of operational acceptance.

Another factor effecting the thermal system automation scope was the removal of all but safety critical FDIR from on-board.

The TAAP has two primary activities: development of a monitoring and diagnosis system for the ATCS and development of a faultable high-fidelity simulation (HFS) of the ATCS. While the HFS is not the main subject of this paper, it is crucial to the overall success of the KBS development. The high-fidelity simulator will have interactive capabilities and provide dynamic bus performance to simulate known ATCS fault modes. Development and testing of FDIR software requires faulting the hardware in various ways while observing and tuning the detection and isolation process. This type of hardware testing is extremely expensive. Faults which might damage the ATCS must obviously be avoided, yet a method is needed for testing all conceivable faults. The HFS will allow inexpensive testing of virtually any fault without risk to hardware.

The TAAP KBS is a hybrid design approach using a combination of conventional programming, tables, rule-based technology and model-based reasoning to provide powerful and flexible diagnositic expertise for the ATCS. TAAP will initially act as an extension to the SSF baseline FDIR capabilities, and could be migrated on-board in stages as additional computational resources are made available. The top level fault isolation process used by the TAAP KBS is:

1) The KBS determines that a sensor has passed one of its alarm limits;
2) A table-based check is performed to quickly catch a small number of potentially catastrophic failures;
3) The KBS uses local propagation and consistency checking in a model to determine whether the sensor reading is valid or the result of faulty instrumentation;
4) If the alarm is valid, the KBS uses a rule-based system to isolate faults defined in the Failure Modes and Effects Analysis (FMEA) [FDIR 88];
5) If the rule system is unable to isolate the fault conclusively, then control is returned to a model-based reasoning (MBR) algorithm for further diagnosis.

Part of the system is implemented using CLIPS v5, which includes an object system called COOL (CLIPS Object Oriented Language). CLIPS is primarily a rule-based forward-chaining inferencing tool. The TAAP rule-based system consists of forward-chaining rules driven by ATCS sensor data and calculated values. The COOL environment is different in some ways from a pure object oriented programming (OOP) language, however it does have the primary characteristics that an OOP system must possess: abstraction, encapsulation, inheritance, polymorphism and dynamic binding. This allows us to implement a "functional simulation" of the ATCS by defining objects and relationships in the COOL environment.

Procedural and numeric-intensive portions of the KBS are currently implemented in structured C code. This includes interface functions needed to handle input and output of parameters to external functions, as well as some calculations involved in model propagation.

Figure 3 shows a functional view of the TAAP KBS architecture. Real-time sensor
data from the HFS or actual hardware is accepted into the KBS by Data Queue code which stores time-slices of data and performs several checks. It checks flags for each sensor to see which values have been updated, then it creates a Suspect Sensor List (SSL). The SSL contains information about which sensor values have exceeded KBS alarm limits. The Data Monitor module is primarily responsible for standard calculations performed at the end of each time-slice. These include: recalculating/updating pseudo values, updating short- and long-term trends, checking/updating historical deltas, maintaining current-value-duration information, and determining qualitative mappings. The Data Queue and Data Monitor together contain a representation of the current system configuration at any time (i.e. valve states, switches, power, etc.).

The KBS Controller is the primary switching center for the KBS. First, it inspects the SSL to determine if sensor validation is required. If necessary, it then supplies the model with sensor readings corresponding to model slots and invokes sensor validation. It also builds facts and asserts them into the Rule-Based System (RBS) facts list. When the RBS or model-based sensor validation is complete, the KBS Controller interprets their diagnoses. If the instrumentation is deemed valid and the RBS cannot match against a known fault, then the Controller initiates model-based reasoning again, this time for the purpose of component fault diagnosis.

Sensor validation is performed when there is exactly one suspect sensor. When more than one sensor is referenced by the SSL, sensor validation is not performed. The assumption is that if only one sensor is out of its application limits, then it is more reasonable to suspect that the sensor has failed than to suspect that an ATCS component failure is causing a single perturbation. After the model has been instantiated with current sensor values, pre-compiled hierarchical information (generated directly from the model) is used to determine the scope of propagation required. The model is then propagated to bind state variables not connected to sensor instrumentation and to compute an expected value for the suspect sensor. The observed readings at each point are then compared with the corresponding computed values in the model. If these two values are within tolerance for all non-suspect sensors, then the suspect sensor is diagnosed as invalid. If one or more non-suspect sensors are out of tolerance (reading vs. computed value), then no determination can be made regarding the validity of the suspect sensor. More specifically, we now have reason to believe that our initial assumption of nominal operation was incorrect.

Note that the model is propagated with the assumption that the bus is operating nominally. In fact both nominal and off-nominal operating conditions cover continuous ranges. For example, if one evaporator has been shut down, the reconfigured bus can still be viewed as operating nominally. Conversely, there is no discrete distinction between off-nominal behaviors. An infinite number of scenarios are possible.

The RBS uses forward chaining rules, matching on patterns of system status information. The design intention of the RBS was to represent the thermal expertise at the highest possible level in the rules. Underlying functionality required for this purpose is generally implemented in the faster procedural language - C. The following is an example of the left-hand side of a rule showing the use of high level thermal knowledge.

```
(defrule ...
   (st-trend TOTAL_HEAT_LOAD steady)
   (qpseudo SUBCOOLING very_low)
   (qsensor BPS702 nominal)
   (test
     (> (extrap BPS702 20.0 STT) 300))
   ... )
```

Referencing qualitative states (e.g. steady, very_low, nominal) rather than numbers in rule premises makes the

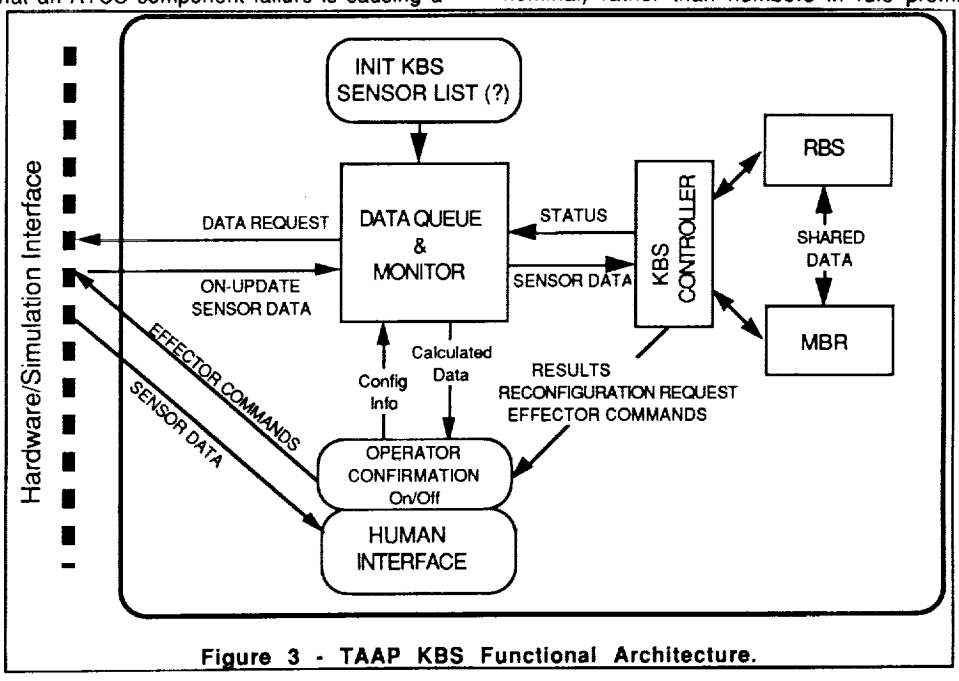


Figure 3 - TAAP KBS Functional Architecture.

RBS much more robust. Transition points are defined and can be modified such that a quantitative value range corresponding to a qualitative translation of "low" may be different depending on the current system mode. This allows the rules to represent thermal knowledge the way it is expressed by the thermal engineer. A straight-forward statement such as "determine if the heat load is steady" represents a significant amount of knowledge, non-trivial calculation and data tracking. The fundamentally numeric-intensive portions are handled procedurally while only the diagnostic knowledge is placed into the rules. Another benefit of qualitative techniques is that pre-compiled relationships (extracted directly from the model) can be used to replace primary sensor values with secondary choices. On-orbit a failed sensor may not be replaceable for several months. Working with the instrumentation available is imperative.

Model-based reasoning for component diagnosis is the most complex and "cutting-edge" portion of the KBS. While the model-based sensor validation phase could assume nominal operation, model-based component diagnosis is much more difficult. The MBR component fault diagnosis makes heavy use of thermal specific knowledge for hypothesis generation and validation.

In performing component diagnosis the entire model is propagated. This will prevent missing systemic faults, where failures in one part of the bus quickly impact a structurally distant portion. The assumption is made that the bus is operating normally, and that all of our sensor instrumentation is valid. Propagating the model and comparing sensor readings to computed values allows us to build a fault-symptom list. The fault-symptom list represents inconsistencies between observed and computed values.

The fault-symptom list is then "mapped" onto a causal relationship model of the ATCS. The causal description of the system is used to generate hypotheses of component failures which could be causing each of the symptoms. For each inconsistency represented in the fault-symptom list the causal model will determine single-component failures which could explain that specific inconsistency. The intersection of the component-failure hypotheses sets returns only those which could be responsible for all the symptoms. A future capability to diagnose multiple simultaneous independent failures would be added by expanding the scope/knowledge represented in the causal model. Due to reliability requirements, diagnosing simultaneous independent failures is not considered a primary goal of this project.

The hypotheses generation phase will result in hypotheses which could account for all the observed symptoms. The validation phase will determine which of these can also account for the remaining observations. Each failure hypothesis is simulated in the faultable, quantitative model. For example, if we hypothesize that a blockage just downstream of the BPRV could cause all the inconsistencies, then we reconfigure the model to reflect a valve at that location being closed and run a simulation. If one of the simulations results in state values comparable to current observations from the sensors, then that hypothesis has been validated (i.e., any surviving hypothesis represents a fault). If none of the hypotheses are supported by simulation, then further discrimination between them may not be possible. Prompting the human operator for more information might allow us to discriminate further between hypotheses. However, automatically determining the appropriate questions depending on the situtation is a difficult task not currently within the scope of this project.

The multi-stage FDIR process just described makes significant additions to the baseline approach. The design attempts to use the best parts of several methodologies while avoiding their restrictions.

The functional use of a Data Queue and Monitor for representation of the current system configuration has advantages over both the baseline and traditional model-based approaches. It provides the structure and data access necessary for complex procedural control processes to be kept updated while avoiding the overhead of conventional OOP systems. Since data control, monitoring and distribution are core requirements of every KBS function, the optimization of these activities is crucial to obtaining desireable performance.

In an early stage, the TAAP KBS uses a minimal table lookup for detecting safety-critical faults, similar to the baseline approach. The TAAP approach allows even the safety critical routine to utilize secondary sensor-source information generated from the model. Just as in the RBS, this makes the table lookup more robust.

The use of a KBS Controller as a centralized switching mechanism for reasoning accomodates a hybrid, multiple technique approach, and effectively modularizes the entire reasoning portion of the system. The relative maturity of rule-based reasoning means the RBS would be more likely to migrate on-board earlier than the MBR capabilities. The decoupling of diagnostic approaches has an advantage of allowing initial ground-based support versions to easily run on seperate, networked workstations if necessary. This improves the prospects of achieving satisfactory response times.

By assuming that multiple simultaneous sensors failures are extremely unlikely, the need for an ongoing sensor validation routine is avoided. When sensor validation is needed, a single propagation and straight-forward consistency check are sufficient.

As mentioned above, some variation of the RBS will likely be the first ATCS advanced automation to migrate on-board. Advantages of the rule-based system over the baseline table-driven approach are numerous. The rules represent diagnostic knowledge at a high level, allowing easier human interpretation, maintenance and meaningful explanation capabilities. The RBS is not tied to a specific configuration. Its data-driven nature combined with support code for primary and alternate instrumentation allows it to degrade more gracefully than a table lookup.

Many previous automation efforts have been heavily driven by schedule, resulting in technical design decisions that range from carefully considered to poorly designed. The fact that "deep reasoning" models can be built from design activities rather early in a program allows us to have begun implementation of this KBS long before first element launch, or even completion of the thermal testbed work. The TAAP model-based reasoning for component diagnosis will require several test and modification cycles before the system is fully functional. Before first element launch the thermal system hardware will go through at least three more major milestones representing potential design changes. All of this serves to further point out the advantages of a model-based approach.

The device oriented representation used for MBR is the cognitive link between software and hardware. By keeping development responsibility for the KBS within the thermal systems group, device models for reasoning can be updated in a timely and accurate manner as new design information becomes available. It is clearly advantageous to have the KBS developers located with the ATCS system engineers.

Conclusion

The table lookup approach described fits well with current onboard constraints of memory and processing power. Although the tables will be difficult to maintain and expand, the inclusion of only safety-critical faults may keep them small enough to avoid any significant problems. Failure of onboard sensor instrumentation can cause problems for the table driven FDIR. Temporary changes in system configuration, such as isolating and shutting down a single evaporator, may require table changes. Finally, the continuous nature of the ATCS when operating nominally may expose the biggest drawback of the table lookup approach.

A hybrid KBS approach to advanced automation will provide both lower lifecycle costs and increased capability. The high-level knowledge representation used in the KBS allows changes to be made easily and new maintenance and support personnel to come up to speed quickly. The KBS will be more robust in reasoning over a degraded or reconfigured system than a table driven approach. The diagnosis of off-nominal behavior, not just known faults, will be possible with the hybrid approach described. The KBS will initially act as a ground-based extension to the core capabilities provided by the onboard table lookup.

The combination of stable, mature technologies with state-of-the-art MBR approaches is advantageous for several reasons. It allows portions of the KBS to be ported on-board as they are completed, confidence is developed, and computer resources become available. It is advantageous to the FDIR development effort to use existing resources and technology. The forward chaining rules used in the RBS, for example, are considered very mature technology. Each of the approaches require successively more speed and RAM for actual implementation.

Given significant mistrust of artificial intelligence techniques by the thermal engineers, user confidence in a system such as the TAAP KBS must be earned gradually. A system offering only a small improvement over baseline monitoring and control capabilities would have difficulty gaining acceptance. The TAAP approach will allow an evolution of functionality and in so doing, will enable engineers to become more confident is this technology. New technologies are being developed all the time and the TAAP approach will allow integration of such technologies as they become available (e.g. predictive maintenance, faster MBR systems, etc.).

The cost of automation endeavors has often been excessive and in some cases prohibitive. Viewed more closely, however, a KBS approach to automation is cost effective. The development of the TAAP KBS for FDIR also supports a console operator in real-time and has components in its implementation (both knowledge bases and inferencing techniques) useful in predictive maintenance systems, training, and autonomous operations. If the common underlying system knowledge can be used in deploying each application then additional knowledge acquisition costs are saved. If these factors are used to effectively prorate project costs, the apparent high price of automation is reduced.

The cost of advanced automation can also be compared to the cost of extensive human training for monitoring of space systems. The full integrated mission simulations used for training shuttle support personnel cost thousands of dollars per day. These training costs alone demand that any task which can be effectively automated, should be automated.

In a major multi-year, multi-national development effort such as the SSFP, many design changes for various reasons will be made, impacting all aspects of the project. These design changes also have serious impact on development of FDIR capabilities for effected systems. A dynamically changing hardware environment favors the use of a hybrid approach with a device oriented model-based representation over the baseline techniques described.

Acknowledgements

TAAP KBS design and development reflects the efforts of numerous people at MDSSC and NASA. Substantial contributions were made by: Roy Steel, William Morris, Charlie Robertson, Jerry Snider, and Sunil Fotedar. MDSSC elements involved are: Thermal Systems Group, Advanced Automation Technology Group (AATG), and Systems Engineering and Integration (SE&I). NASA involvement includes: JSC Crew and Thermal Systems Division, JSC Automation and Robotics Division, and NASA Ames Research Center Advanced Missions Technology Branch.

References

[ATHENA 86]
Idaho National Engineering Laboratory, "ATHENA Code Manual: Code Structure, System Models, and Solution Methods," EGG-RTH-7397, EG&G-Idaho, Inc, September 1986.

[Glass 90]
Glass, B. J., et. al., "TEXSYS: a large scale demonstration of model-based real-time control of a space station subsystem," First International Workshop on Principles of Diagnosis, Stanford University, July 23-25, 1990.

[FDIR 88]
Crew and Thermal Systems Division, "BAC Two-Phase Thermal Control System: FDIR Document," NASA - Johnson Space Center, 1988.

N93-11931

INTELLIGENT DIAGNOSTICS SYSTEMS

Barbara M. McQuiston
Ronald L. De Hoff
Systems Control Technology, Inc.
2300 Geng Road
Palo Alto, CA 94303
(415) 494-2233

Intelligent systems have been applied to today's problems and could also be applied to space operations integrity. One of these systems is the XMAN™ tool designed for "troubleshooting" jet engines. XMAN is the eXpert MAiNtenance tool developed to be an expert information analysis tool which stores trending and diagnostic data on Air Force engines. XMAN operates with a "network topology" which follows a flow chart containing engine management information reports required by the government's technical ordor procedures. With XMAN technology, the user is able to identify engine problems by presenting the assertions of the fault isolation logic and attempting to satisfy individual assertions by referring to the databases created by an engine monitoring system. The troubleshooting process requires interaction between the technician and the computer to acquire new evidence form auxiliary maintenance tests corroboration of analytical results to accurately diagnose equipment malfunctions. This same technology will be required for systems which are functioning in space either with an onboard crew, or with an unmanned system. This paper addresses the technology and lessons learned developing this technology while suggesting definite applications for its use with developing space systems.

BM2-2291.1

Session I3: MONITORING AND CONTROL

Session Chair: Dr. Robert Lea

N93-11932

SPACECRAFT ATTITUDE CONTROL USING A SMART CONTROL SYSTEM

Brian Buckley
Interface & Control Systems
1944 South Dairy Road
West Melbourne, Florida 32904

Louis Wheatcraft
Barrios Technology, Inc.
1331 Gemini Ave
Houston, Texas 77058

Abstract

Traditionally, spacecraft attitude control has been implemented using control loops written in native code for a space hardened processor. The Naval Research Lab has taken this approach during their development of the Attitude Control Electronics (ACE) package. After the system was developed and delivered, NRL decided to explore alternate technologies to accomplish this same task more efficiently. The approach taken by NRL was to implement the ACE control loops using expert systems technologies. The purpose of this effort was to:

- Research capabilities required of an expert system in processing a classic closed-loop control algorithm.
- Research the development environment required to design and test an embedded expert systems environment.
- Research the complexity of design and development of expert systems versus a conventional approach.
- Test the resulting systems against the flight acceptance test software for both response and accuracy.

Two expert systems were selected to implement the control loops. Criteria used for the selection of the expert systems included that they had to run in both embedded systems and ground based environments. Using two different expert systems allowed a comparison of the real-time capabilities, inferencing capabilities, and the ground-based development environment.

The two expert systems chosen for the evaluation were Spacecraft Command Language (SCL), and NEXPERT Object. SCL is a smart control system produced for the Naval Research Lab by Interface and Control Systems (ICS). SCL was developed to be used for real-time command, control, and monitoring of a new generation of spacecraft. NEXPERT Object is a commercially available product developed by Neuron Data.

Results of the effort were evaluated using the ACE test bed. The ACE test bed had been developed and used to test the original flight hardware and software using simulators and flight-like interfaces. The test bed was used for testing the expert systems in a "near-flight" environment.

This paper details the technical approach, the system architecture, the development environments, knowledge base development, and results of this effort.

Introduction

The Naval Research Lab has developed an upper stage used for orbital insertion of satellites. The upper stage is spin stabilized until it reaches the insertion orbit. Once in the desired orbit, the upper stage is spun down and stabilized using momentum wheels and reaction control thrusters. The upper stage then jettisons the spacecraft allowing it to move into its parking orbit.

All aspects of the orbital transfer maneuver are controlled by the Attitude Control Electronics (ACE).

The ACE subsystem is semi-autonomous and can issue thruster commands to maintain the desired attitude. The ACE control loops were developed in the flight processor's native assembly language. The development of the algorithms required years of design, testing and elaborate simulation.

Once the ACE system had been successfully launched, the NRL began exploring alternate technologies for developing the same task. The NRL selected an expert system approach since it is event driven and would allow the ACE system to take advantage of 90's technologies.

The NRL development effort was to focus on the following goals:

- Test expert system technologies to prove productivity could be increased, and the system could be reused for other needs.
- Research the use of expert system technologies to implement real-world closed-loop control algorithms.
- Research the type of development environment that would be required to develop and test an expert system for closed loop control, in contrast to the traditional environment used to develop embedded systems.
- Research the complexity of an expert system design and the difficulty in developing the expert system compared to a traditional approach.
- Compare the performance and the accuracy of the resulting expert system against the proven flight implementation.

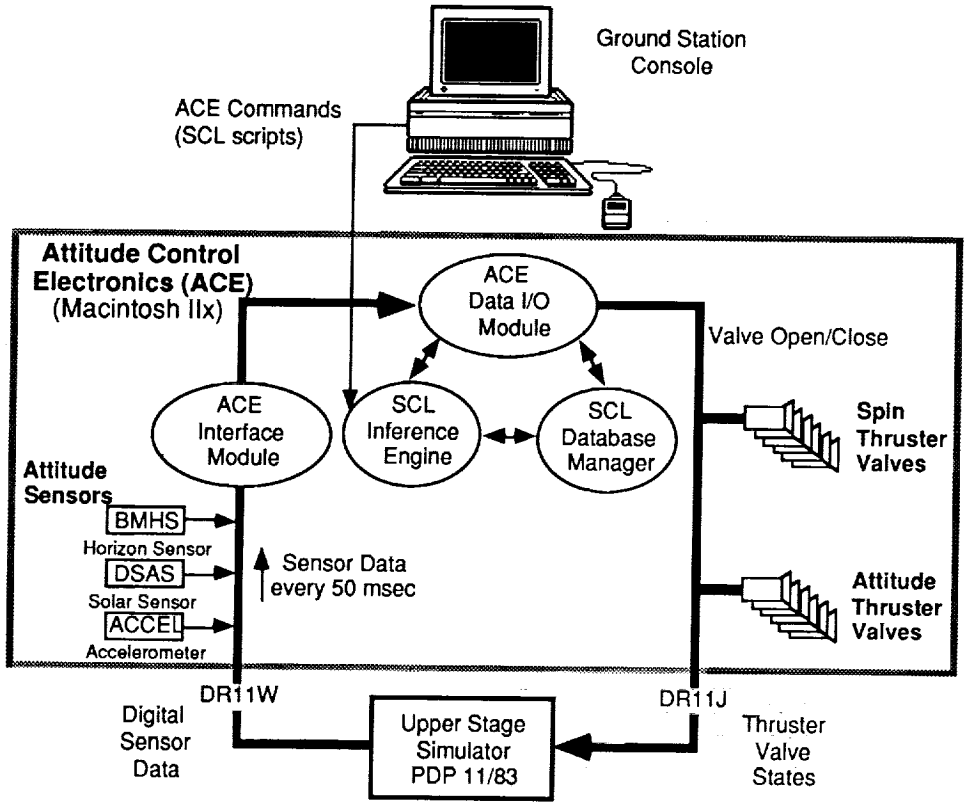
To objectively develop and evaluate the expert system approach, two expert systems were chosen for the development effort. The expert systems that were chosen were required to run in both embedded systems and ground based environments. Both SCL and NEXPERT were chosen since they both were available in an embedded environment. Both systems also have a ground based development environment that is available using an intuitive man-machine interface. The Macintosh II was chosen as the development environment for both expert systems.

Spacecraft Command Language (SCL) is an expert system that was developed for a NRL satellite controller by Interface and Control Systems, Inc. NEXPERT Object is a commercial expert system that was developed by Neuron Data. The two expert systems use radically different approaches to the implementation of the embedded system application. SCL is a total control environment that runs on the embedded processor. The processor, I/O, and operating system specifics are isolated to a small number of low level routines. The SCL system is bound to the operating system specific calls and hardware interfaces using a small amount of "glue" code. The flight algorithms are in the form of scripts and rules and are written in the SCL fifth-generation language. Scripts and rules are compiled on the ground and uploaded to the embedded system where they are interpreted by the SCL real-time executive.

The NEXPERT Object system consists of a library of callable routines that are called from the application code. The application code is written in "C" and makes calls to the NEXPERT library for control of the rule evaluation and inference strategy. The NEXPERT rules are written in a high level language, and are compiled on the ground and interpreted by the on-board target processor.

The NRL contractor that implemented the original ACE flight algorithms was chosen to implement the expert systems in both SCL and NEXPERT. The contractor was familiar with the flight algorithms as well as the flight simulator, and had no contractual ties to either Interface and Control Systems, or Neuron Data.

The ACE flight hardware had been tested against a simulator that gave a three axes model of the spaceborne upper stage. The simulator provided scenarios of normal spacecraft maneuvers as well as variations with anomalies introduced which would require the ACE algorithms to take corrective measures.



SCL ACE System Architecture

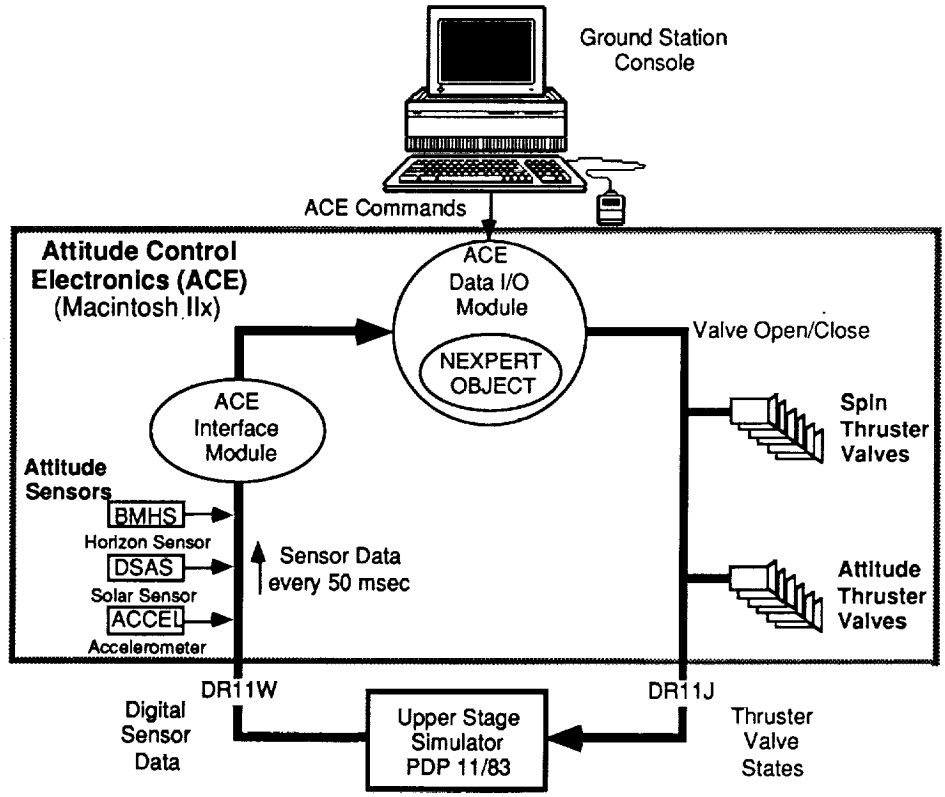
Development Approach

The contractor chose to use the Macintosh II as both the development and run-time environment for the NEXPERT and SCL expert systems. The Macintosh 68000-based machine was roughly equivalent to the flight processor in terms of throughput. The ACE simulator was hosted on a PDP 11/83 and was coupled to the Macintosh using two parallel I/O boards. One board was used to receive sensor data from the simulator, the other board was used to issue commands to initiate thruster maneuvers.

Custom code was developed for both expert systems to support the ACE prototype. The SCL expert system required "glue" code to be written in "C" to couple the bidirectional data between SCL and the I/O boards. The NEXPERT application executive was written in "C". The executive made calls to the NEXPERT library to control the expert system, it also performed tasks that could not easily be implemented using NEXPERT's rules. The NEXPERT data I/O module, which was much the same as that used for SCL, performed all communications with the simulator. Additionally, the I/O module performed timing functions that were not available with NEXPERT.

System Architecture

The ACE simulator emits an 11 word packet of raw sensor data every 10 msecs. A parallel interface was required to ingest the packet, decommutate the sensor data, translate the data into engineering units, and notify the expert system every 10 msecs. The engineering unit form of the sensor data is used by the expert systems for evaluation by rules and scripts. The interface software also stores every fifth packet for attitude filter and nutation calculations.

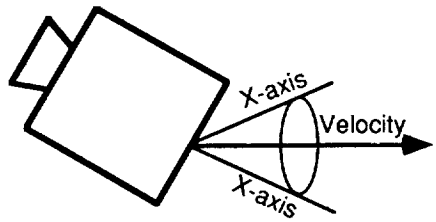


NEXPERT ACE System Architecture

The sensor data is used by the expert systems to perform four closed loop control tasks:

- Spin Rate Control
- Sun Angle Control
- Active Nutation Control
- Active Nutation Filter Calculations

The upper stage does not maintain a constant center of gravity since one or more satellites may be jettisoned. The ACE algorithms must take the changing center of gravity into consideration when making attitude adjustments. The upper stage is normally spin stabilized. Under optimum conditions, the spacecraft spins about its X axis around the velocity vector and both are parallel. Nutation is introduced when the spacecraft begins to wobble. In a simple scenario, one end of the spacecraft begins to wobble as depicted in the drawing to the right. The X axis "cones" about the velocity vector. The cone angle is nominally kept to +/- 0.25° by the ACE algorithms. (The cone angle is discussed later during the evaluation of the systems.) The ACE control loops must determine the appropriate thruster(s) to fire, the exact time of the firing, the duration of the burn, and the number of burns required to correct the nutation.



Upper Stage Nutation

The ACE interface software receives raw sensor data from Digital Solar Aspect Sensors (DSAS) and Body Mounted Horizon Sensors (BMHS). These sensor readings are used to calculate engineering values for the

spin period, sun angle, and active nutation. If the knowledge base determines a corrective action is required, a command is sent to enable the appropriate thruster for some delta time. Corrective actions may be taken to correct either the spin rate, sun angle, or active nutation. All corrections take place based upon commands generated by the ACE algorithms.

Software Development

The accuracy of the thruster maneuvers is critical to correcting the attitude of the spacecraft. The thruster valve open and close times are critical. With precise timing, the number of corrections can be reduced, thus minimizing the amount of reaction control propellant used.

Since NEXPERT had no precision timing capabilities, the functionality had to be provided by writing "C" code in the interface module to implement the capability.

SCL not only provides a language that supports the definition of rules, the language also allows procedural programming using scripts that share a common syntax. The SCL scripts provide a time synchronized execution and are used for the precision timing required for thruster maneuvers.

The ACE prototype also required a method of enabling and disabling various control modes. Again this was handled by SCL scripts, while additional source code was written for NEXPERT to handle this requirement.

Knowledge Base Development

Both expert systems provide a window-based development environment for defining the knowledge base. The NEXPERT system provides a graphical view of the rule heritage, SCL had no graphic representation at the time. The NEXPERT graphics allowed the engineers to view the relationship between database items and rules in the knowledge base. Both systems allow the user to define the knowledge base using a windowed text editor.

The SCL expert system allows the user to mix scripts and rules to form the knowledge base. Scripts may be called by rules using SCL. The SCL scripting feature reduced the complexity of the knowledge base. The english-like scripting capability of SCL allows loops and control structures to be written. These loops and control structures are difficult to construct using rules. Scripting was also used to enable and disable the various control modes. The NEXPERT system required all control algorithms be written as rules and the control functions be implemented by developing additional rules or "C" source code.

The Inference technique of both systems came into play when designing the knowledge base. Both systems are primarily forward chaining expert systems. The NEXPERT inference method employed a prioritized queue for sequencing the execution of rules. The "agenda" allows rules to be dynamically scheduled for execution based on priority. As rules are fired and database items are changed, new rules are merged onto the agenda.

At the time of the ACE project, the SCL inference method used a depth-first strategy. When a database item changes, the rules that reference the item were executed in a prioritized order. If a database item was changed, the inference engine focused on that item and executed rules associated with it before returning to the rules for the previous item. Because of the depth-first method, rules of lower priority could be evaluated before rules of higher priority. This was caused by the system focusing on the last item which changes. Since the ACE prototype was developed, the SCL inference engine has been modified to allow a prioritized queue of rules to be evaluated similar to the agenda in NEXPERT and CLIPS (an expert system developed by NASA Johnson Space Center). The inference strategy for SCL is now user selectable for either

depth first evaluation or evaluation using an agenda.

The SCL inference engine time-shares with the SCL command interpreter, i.e., rules are evaluated in parallel with script execution. This method allows scripts to execute at timed intervals and set database variables that control the mode of the attitude determination tasks. Changes to these variables result in corresponding rules to be executed, allowing timing synchronization between the scripts and rules.

The differing inference strategies between the two systems provided some interesting results. Statistics for rule execution for the two systems were collected. The NEXPERT rule base required fewer rules to be executed because of the prioritized agenda. Although the SCL expert system executed some rules more than once, the end result (reflected in telemetry variables) was identical for both systems. For a given test scenario, the SCL system was able to evaluate more rules per second.

Object Representation

The knowledge base is centered around a related group of objects. Both NEXPERT and SCL require the knowledge engineer to describe the ACE data points for both commands and telemetry in terms of objects. Rules in the knowledge base respond to changes in the values of these objects and may induce changes in one or more other objects.

Both SCL and NEXPERT are generic expert systems, that is, neither have any ties to a given spacecraft. SCL describes telemetry sensors, command actuators, and derived items in a database. Telemetry sensors are physical sensor readings, while derived items are data points that are calculated based on readings from one or more sensors. Command actuators are data points that are used to activate/deactivate relays, or serial data words which are interpreted by local or remote command functions. SCL groups these items by a sorting field called the subsystem.

NEXPERT allows data points to be defined in terms of classes, subclasses, objects, subobjects, and properties.

The SCL database was designed to support a real-time system and was purposely designed to have a limited amount of data abstraction in terms of object oriented programming. The tradeoff was felt to be necessary due to memory and real-time considerations when running in an embedded environment.

Testing the Knowledge Base

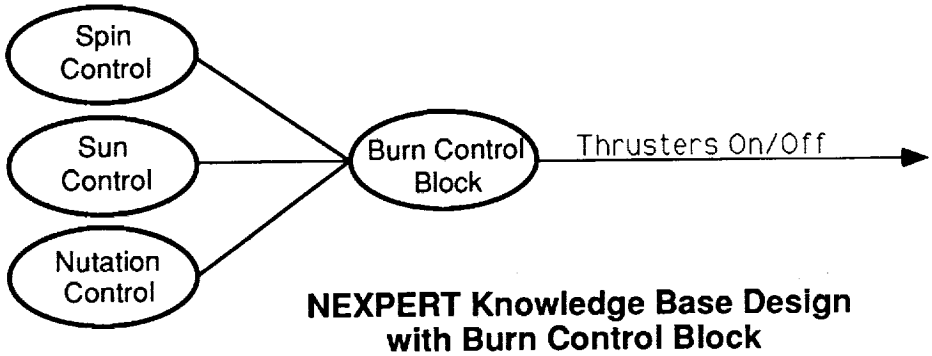
Once the knowledge base was developed, the rules and scripts must be tested and debugged. The NEXPERT system provides a debugger similar to those used in conventional programming systems. Break points may be set on rules and data objects contained in the knowledge base. When a break point is encountered, data objects may be examined and/or modified.

SCL allows the user to examine and modify the knowledge base data points from its command window. Much of the SCL syntax that is used in scripts and rules is also valid from the command window. To implement a break point, a "stop inference engine" statement needed to be placed in the script or rule that a break point is desired. A full featured source level debugger is currently under development for a future release of SCL.

SCL also allows the user to trace all script and rule execution or individual scripts and rules may be selected for tracing. Several levels of tracing are supported, as well as tracing on all or selected database points.

Expert System Verification

The resulting expert systems were tested for response and accuracy and compared to the actual flight software. To verify the results, the ACE flight software dynamic model was used in the same manner as it was to acceptance tests the flight software. The following tests were used to



NEXPERT Knowledge Base Design with Burn Control Block

verify the closed-loop control algorithms for spin period, sun angle, and active nutation:

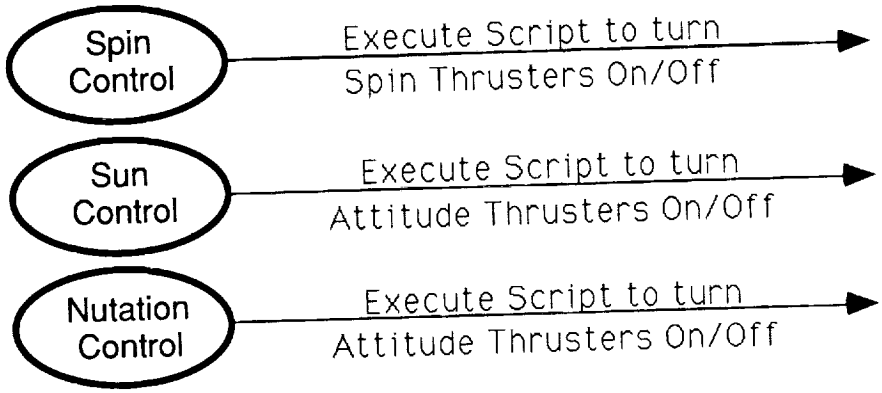
- Sun Angle Correction - begin at 75° angle, maintain to 90° +/- 10°.
- Sun Angle Correction - begin at 105° angle, maintain to 90° +/- 10°.
- Spin Period Correction - begin at 3.5 r.p.m., maintain 5.0 r.p.m. +/- 10%.
- Spin Period Correction - begin at 5.5 r.p.m., maintain 5.0 r.p.m. +/- 10%.
- Nutation Correction - begin at 0.1° cone angle, maintain 0.25° cone angle.
- Nutation Correction - begin at 10.0° cone angle, maintain 0.25° cone angle.
- Nutation and Sun Angle Correction.
- Spin Period and Sun Angle Correction.
- Spin Period, Nutation, and Sun Angle Correction.

The closed loop control of the spin period and sun angle worked equally well using both expert systems. Both returned the same results as those reported by the ACE flight software. While testing the nutation control with NEXPERT, problems were detected while attempting to initiate a thruster maneuver upon encountering a zero-crossing in the nutation filter. The problem was traced to the design of the NEXPERT knowledge base and the amount of time required to process the rules upon encountering a zero crossing. Because of this, the NEXPERT knowledge base was not able to correct the nutation.

To correct this problem, the NEXPERT knowledge base was modified to reduce the number of rules that needed to be evaluated at the zero crossing. This allowed the active nutation to be corrected to the desired level. The NEXPERT knowledge base corrected the low nutation as efficiently as the ACE flight software, however, an additional 400 millisecond burn was required of the 30 lb. thruster pair to correct the 10° cone angle.

Having seen the difficulties in achieving accurate timing for thruster burn start and stop, the knowledge base architecture was redesigned to take advantage of the SCL scripting capability. Previously using NEXPERT, the spin control algorithm, the sun control algorithm and the nutation control algorithm fed the burn control block. The burn control block controlled the firing of all thrusters. The SCL knowledge base was redesigned to allow rules that control the spin, sun, and nutation control and to execute scripts directly which initiate and terminate the appropriate thrusters.

This design allowed the SCL knowledge base to correct all nutation as effectively as the actual ACE flight software. The NEXPERT knowledge based could also have been redesigned, but it was not deemed to be sufficiently beneficial.



SCL Knowledge Base Design with Scripts

Conclusions

The ACE expert systems proved that expert system technology can be applied to classic control loop algorithms. The speed, accuracy, and memory requirements can be met using a more modern approach. However, a commercial product was not deemed feasible. The extensive memory usage and real-time limitations of the commercial product made it a poor choice for this application. NEXPERT's inability to keep pace with the control loops made it an unacceptable alternative.

SCL proved to be a capable of performing all ACE monitor and control functions at least as well as the actual flight code. SCL is a specialized product designed for satellite command and control systems. Since SCL was designed to run in real-time embedded environments, overall memory usage has been minimized, and SCL does not rely on dynamic memory allocation (since it will eventually cause fragmentation.) SCL was designed to run on spacecraft with radiation-hardened processors in the 1.5-3 MIPS range and tight memory requirements. With the skyrocketing cost of rad-hard error correcting memory, the SCL designers chose not to include more extensive object-oriented features.

Because of the favorable results of the ACE effort, the SCL expert system was chosen as the embedded flight controller software for a future NRL satellite to be used on a national program. SCL will be used for the satellite as well as the dispenser. SCL has also been chosen as the ground based expert system at NRL ground stations to support upcoming missions as well as provide advisory systems for existing satellite systems.

The expert system approach allows much greater productivity for the engineers since they are working in a very high level language and have sophisticated development tools available. The systems can also be modeled and tested on desktop workstations rather than having to use specialized test fixtures.

Because SCL is not a mass-marketed commercial expert system, changes could be made to the SCL system based on lessons learned during the ACE project:

- The agenda inference strategy was added.
- A source-level debugger is under development.
- The SCL development Man Machine Interface is being ported to Motif/X-Windows to run on many popular workstations and X-terminals.

- A graphics interface for SCL is being developed to allow the developer to view relationships between database objects, rules, and scripts.
- Several extensions to the SCL syntax were requested and have been added to the system.

Applied Expert System Technology

In the past, expert system technology has been difficult to use, expensive, and ran only on specialized hardware. Modern expert systems are available on a wide variety of processors and have become an efficient and cost-effective solution for systems development. The expert system technology can easily be merged with conventional technology to allow simplification of system development. By developing realistic dynamic simulations, rapid prototyping and modeling of subsystems and entire systems is practical. Developers can checkout complex systems in a desktop environment and can find potential problems early in the development cycle. These simulations can be part of a test bed that will not only support control system software development and validation but the same system can be used for training. This expert system technology can be merged with other types of technology such as databases, spreadsheets, graphics, hypermedia, animation, sound, and conventional code.

Expert systems can also aid in the design and development of systems by providing a reasoning mechanism that models human reasoning, providing data representation that more closely models the "real world", and allowing generic systems to be developed and be reused, while localizing the application specifics in the knowledge base.

SCL

SCL is unique in the area of command and control because of its small size, ease of use, versatility, reusability, and adaptability. SCL offers, in one package, the following capabilities:

- Real-time command and control - not a static query expert system (Medical, Financial, etc.).
- Complete integrated environment with re-usable software allows SCL to be used for diverse applications.
- Supports embedded and distributed applications.
- Scripting capability for real-time command and control, monitoring, modeling, simulations, and training.
- Easy to learn, excellent Man/Machine Interface (MMI).

Compact sizing as well as the abstraction of hardware specifics allows the standard SCL kernel to be used on a variety of hosts for embedded processors as well as mini and micro-computer and workstation applications.

SCL can be used for software simulations of subsystems as well as an entire spacecraft, systems modeling, and training.

DOD, NASA, and the commercial sector will be able to use SCL for any applications requiring smart subsystem control and monitoring. These applications could include uses in support of the Space Station, robotics, future Lunar and Mars missions, as well as commercial applications requiring real-time smart control system process monitoring (petrochemicals, manufacturing, utilities, etc.).

Acknowledgments

We would like to thank the following people for their contributions to this paper: Dave Schriftman, Jim Van Gaasbeck, Patrice Cappelaere, and Dean Oswald.

References

1. Buckley, Brian and Wheatcraft, Louis: "Spacecraft Command Language - A Smart Control System", Interface and Control Systems, Melbourne, Fl., Barrios Technology, Houston, TX., March 1991

2. Van Gaasbeck, James: "Technical Overview of the Spacecraft Command Language" Naval Research Laboratory, Washington D.C., 1991

3. Interface and Control Systems: "SCL User's Guide", Naval Research Laboratory, Washington D.C., 1990

4. Buckley, Brian and Wheatcraft, Louis: "Rapid Prototyping of a Spacecraft Controller", JAIPCC Symposium, Houston Tx., March 1991

5. Buckley, Brian and Wheatcraft, Louis: "Spacecraft Simulations with a re-usable Smart Control System", JAIPCC Symposium, Houston Tx., March 1991

N93-11933

Design Development of a Neural Network-based Telemetry Monitor

Michael F. Lembeck, Ph.D.
Space Industries, Inc.
711 W. Bay Area Blvd., Suite 320
Webster, TX 77598
(713) 338-2676

Abstract

This paper identifies the requirements and describes an architectural framework for an artificial neural network-based system that is capable of fulfilling monitoring and control requirements of future aerospace missions. Incorporated into this framework are a newly developed training algorithm and the concept of cooperative network architectures. The feasibility of such an approach is demonstrated for its ability to identify faults in low frequency waveforms.

1.0 Introduction

As aerospace systems become more complex, the need to quickly predict, identify, and correct faults becomes more critical to mission success. Future systems on board spacecraft will have requirements for longer lifetimes, higher reliabilities, and lower maintenance than previously encountered. To meet these requirements, Artificial Intelligence (AI) tools can assist human operators in anticipating failures in equipment from trends in sensed data. This paper examines the utility of artificial neural networks and the architectures and implementations required for monitoring real-time sensor telemetry signals.

Several methods are available for training neural networks to perform pattern recognition tasks. The Self-Scaling Conjugate Gradient method presented here is shown to be applicable for training neural networks to solve the telemetry signal monitoring problem. This method also demonstrates exceptional performance, measured in terms of network convergence time, when compared to other available training methods.

To monitor signal waveforms and generate an output indicating waveform "health" four "cooperative" neural networks are used. These networks, each monitoring a specific "part" of the signal of interest, are shown to be capable of detecting faults in low frequency waveforms.

1.1 Problem definition

A high priority in the design of real-time monitoring systems is the reduction of large amounts of sensor telemetry data into an immediately useable form for human operators and machine interpretation. Serial instruction-based computation devices monitoring simple limits on telemetry data will be replaced with machines capable of parallel operations which are more adept at pattern identification. Artificial neural networks, which have been shown to be quite adept at identifying patterns in data will play a large role in this effort.[13,14,16]

Expert systems have been developed to perform "intelligent" control or management of system configurations (i.e. the optimum scheduling, switching, or control of devices, given a set of resource constraints and available modes).[4,7] Sensor data indicating the current state of the system is tested against knowledge gathered from experts familiar with the given system's operation. This knowledge may take the form of rules specifying how the system should be configured for nominal operations as well as in the event of a system fault.

Unfortunately, conventional expert systems, running on today's hardware, cannot respond in real-time (defined here as on the order of microseconds to milliseconds) to system faults which require detailed analysis and immediate reconfiguration.[2,20] For such applications, a network can be designed to recognize specific sensor data patterns, associate them with a specific output (much like the assertion of a "consequent" of a conventional rule in an expert system) and

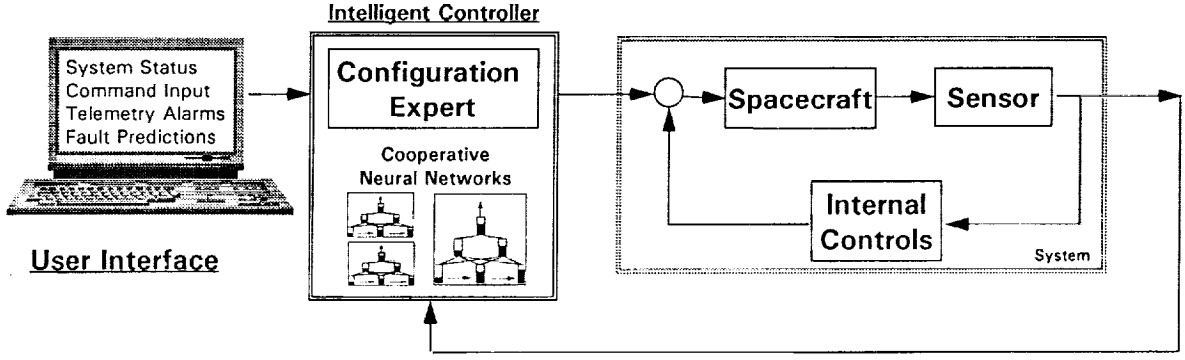


Figure 1. Neural Network-based telemetry monitor reference system.

output the result to some configuration manager or controller interface. Implemented in hardware, neural network response times may approach those required for real-time control.[10,19]

The reference system illustrated in Figure 1 serves as a model for developing and evaluating applications of artificial neural networks to monitoring and control problems. An external configuration manager commands the system to achieve the desired result. Sensors determine the actions taken by the system in response to the controlling input. Additional internal controls may be employed to achieve localized regulation of the system. Sensor data is also reported to an external telemetry monitor (perhaps viewed by a human operator) and routed back to the configuration manager.

2.0 Artificial Neural Networks

An artificial neural network can be defined as a "parallel interconnected network of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way that biological systems do."[9] The simplest model of an artificial neuron is drawn in Figure 2.

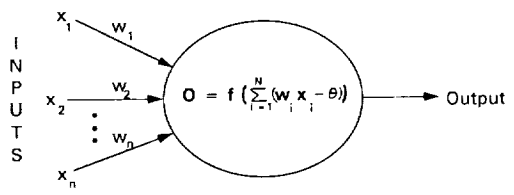


Figure 2. Simplified artificial neuron.

The illustrated simplified artificial neuron receives an arbitrary number of input signals, x_i, each weighted or modulated by a gain, w_i. These weighted signals are then summed and characterized by an interior threshold or bias, θ. Finally, the result is passed through a mapping function, f, to generate an output signal, O. The mapping function generally represents some continuous nonlinear (hard limiter, threshold logic, or sigmoid) "activation" function through which the weighted sum is passed before being expressed as an output.

Collections of connected neurons, making up a single layer neural network, can be constructed by connecting several inputs to more than one neuron. It is customary to index connection weights as w_{ij}, meaning the weight value on the connection from node *i* in a lower layer to node *j* in an upper layer.

In the single layer case, the output functions, O_k, may be computed as:

$$O_k = f_k(\sum w_{ij} x_i - \theta_k) \qquad 1$$

Network architectures composed of several layers of nodes are possible. Hornik has shown that these multilayer, feedforward networks can be used as universal approximators for various functions.[8] Cybenko has shown that arbitrary decision regions (stored patterns) can be arbitrarily well approximated by a continuous feedforward neural network with only a single hidden layer and any continuous sigmoidal ($f(x) = 1/(1 + \exp(-x))$) mapping function.[1] This important result is the justification for concentrating on the so-called "three layer feedforward network."

In this type of network, input signals are passed through an input layer which scales the signals into the range $0.0 \leq x \leq 1.0$ for use by the network. The scaled signals are multiplied by connective weights and input to a middle or "hidden" layer. At each node all weighted values are summed and passed through a sigmoid nonlinearity for output to the next layer. The process is then repeated in the next highest layer of the network in order to produce the desired output.

2.1 Training Neural Networks

Patterns may be stored in a network by varying the connective weights between network nodes until the desired associative outputs are generated in response to the presentation of a training input pattern. Methods by which the weights are iteratively updated are called "training" algorithms. Introductory descriptions of these algorithms and several other neuron models and network architectures can be found in the literature.[9,12]

The training of an artificial neural network to properly generalize and associate a desired output pattern when presented with a given input pattern can be posed as a multivariable optimization problem based on the "generalized delta rule."[17] First, an objective or error function must be defined which represents how well the network has learned its task and is generally a function of the target and actual output values. The optimization objective is to find a set of network weights which minimizes the error function for all pattern presentations.

One way of accomplishing this is to minimize the sum of the squares of the difference between output node values and the desired target values for each output node. In the following expressions, *k* refers to nodes in the output layer, *j* refers to nodes in the hidden layer, *i* refers to nodes in the input layer, and *p* refers to the pattern presentation sequence number. For each input pattern/output target pair, Equation 2 defines a common variation of an error function, E_p, used as a starting point for defining training algorithms for a neural networks. Here t_{pk} is the *p*'th target for the *k*'th node and o_{pk} is the output for that node.

$$E_p = \sum_{k=1}^{K} (t_{pk} - o_{pk})^2 \qquad 2$$

A ***"presentation"*** or ***"function evaluation"*** takes place when a single pattern is presented to a network's input nodes and this input is then fed forward through the network to produce an output at the output nodes. ***"Gradient"*** or ***"derivative evaluations"*** are computed when the error resulting from this presentation is fed-back through the network and weight function gradients are calculated for each weight and bias. After all patterns in a training set have been presented as inputs to a network, a ***"step"*** or ***"epoch"*** is then said to have passed. Depending on the training algorithm, ***"weight updates,"*** or the modification of the connecting weights, may take place after a step or epoch.

2.2 Training Methods

Given the definition of an error function, an expression for incrementally updating the connection weights to minimize this function can be derived. The weight update is formulated as the

derivative of the error with respect to each weight and proportional to some negative constant as given by Equation 3.

$$\Delta_p w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}} \qquad 3$$

Various approaches have been taken to this optimization problem. Some of the most successful, and a new more efficient method, are explained below.

2.3 On-line Back-propagation

On-line back-propagation is the name given to the scheme where a weight update is computed after every individual pattern is presented to the network input nodes and the error (Equation 2) for each pattern is minimized. Thus, one function evaluation, one gradient evaluation, and one weight update occurs with every pattern presentation. While not a true descent method, this robust algorithm has been quite successful in solving a large variety of problems.[17,3] Equation 4 illustrates a common weight update formula used for the hidden-to-output layer connections.

$$\Delta w_{kj}(n+1) = \eta\,(t_{pk} - o_{pk})\, f'(net_{pk})\, o_{pj} + \alpha\, \Delta w_{ji}(n) \qquad 4$$

Here p is the pattern number, t_{pk} is the target value for a given output neuron, o_{pk} is the neuron's actual output, f' is the derivative of the sigmoid function, and net_{pk} is the sum of all inputs to the neuron coming from the hidden layer. The training rate, η, and the momentum coefficient, α, are fixed for the duration of the training session. "Optimal" values of these two constants are usually determined in an empirical manner for each problem.

Aside from its successes, this method has several drawbacks, especially when applied to problems of large scale. Since η is fixed, too large or too small a step may be taken in the descent direction resulting in a violation of the descent condition or an overshooting of the minimum. In the case of back-propagation with momentum, a fixed α may result in a step being taken in a non-descent direction where any η may result in an increase in the system error. Finally, because a weight update is performed after each pattern is presented to the network, the pattern presentation order can have an effect on the rate of convergence.

2.4 Conjugate Gradient Methods

Conjugate gradient methods have a long history of solving large dimensional problems where other methods fail.[5,18] For the neural network training problem it is desired to iteratively minimize the network error function after each set of patterns has been presented to the network. New values for each connective weight w then are defined in terms of the gradient, g_k, and the search direction, d_k at each step:

$$w_{k+1} = w_k + \alpha_k d_k \qquad 5$$

where

$$d_{k+1} = -g_{k+1} + \beta_k d_k \qquad 6$$

$$\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{(g_{k+1} - g_k)^T d_k} \qquad 7$$

The choice for α_k is made by finding the minimum value of the error function through successive line searches (finding the minimum of a function along search direction, d_k). When m is the number of iterations required to locate a minimum to a given tolerance and p is the number of patterns in the training set, each line minimization will require mp function evaluations. After a suitable minimum has been found, p number of gradient evaluations are then required in order to compute the next search direction.

This study was concerned with methods in which storage and computational requirements are minimized for possible onboard spacecraft applications. Algorithms were considered if only the last search direction and gradient need to be retained from step to step and the overhead for computing iterative search directions is low. Higher order methods (BFGS, quasi-Newton, etc.), require storage of additional arrays in order to build up approximations to the Hessian and sometimes incur a significant overhead in search direction computation.[18]

The primary objective here is to find a method which will allow a neural network to reliably converge to a weight set with a minimum amount of function and gradient calculations. These evaluations are the means by which competing algorithms are judged.

2.5 Self-Scaling Conjugate Gradient Method

In his paper on conjugate gradient methods with inexact searches, Shanno reviews several different conjugate-gradient-type methods for application to several classes of problems.[18] One such method, Shanno's Equation 26a, derived from work carried out by Oren and Spedicato, represents a Self-Scaling Conjugate Gradient (SSCG) algorithm which does not require the storage of additional arrays.[15]

With line minimizations performed as before to find the step length, the new search direction for each iteration is given by Equation 8.

$$d_{k+1} = -\frac{p_k^T y_k}{y_k^T y_k} g_{k+1} - \left(2\frac{p_k^T g_{k+1}}{p_k^T y_k} - \frac{y_k^T g_{k+1}}{y_k^T y_k}\right) p_k + \frac{p_k^T g_{k+1}}{y_k^T y_k} y_k \qquad 8$$

where

$$p_k = \alpha_k d_k \quad \text{and} \quad y_k = g_{k+1} - g_k$$

The SSCG algorithm represents a potentially powerful method for carrying out neural network weight-updating training algorithms. This gradient descent method is less susceptible to certain local minimums. Finally, the order of presentation of the input patterns is not critical.

3.0 Temporal Windowing

One approach to formatting telemetry data for input to a neural network is to sample the input signal at discrete time intervals, scale this into a range acceptable by the network, and feed this signal into the input nodes of the network. If the input layer contains more than one node, successive input nodes may be stimulated with time-delayed samples of the input data. Such an approach is called "Temporal Windowing."[6]

Figure 3 shows an example of three temporal windows of a 1.0 Hz sine wave sampled every 100 ms (10 Hz sampling rate).

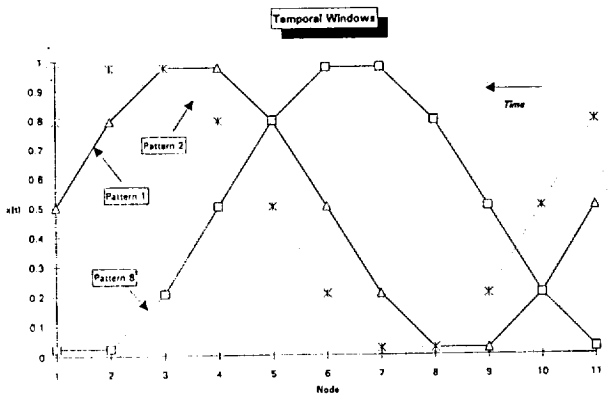


Figure 3. Temporal windows for a 1 Hz sine wave.

Pattern 1 shows the initial snapshot of 11 values making up one complete period of the wave. Pattern 2 represents the next available snapshot, taken 100 ms later. The rightmost data point represents the most recently sampled part of the sine wave. Pattern 8 represents the state of the wave 700 ms after the initial snapshot (Pattern 1). Note that ten patterns are required to show one complete period of the sine wave.

Another way to view temporal windowing is that a sample is taken, displayed to the rightmost input node for a short delay period (100 ms), and then displayed to the node immediately left of the previous node. Figure 4 illustrates this sample, hold, and shift input procedure.

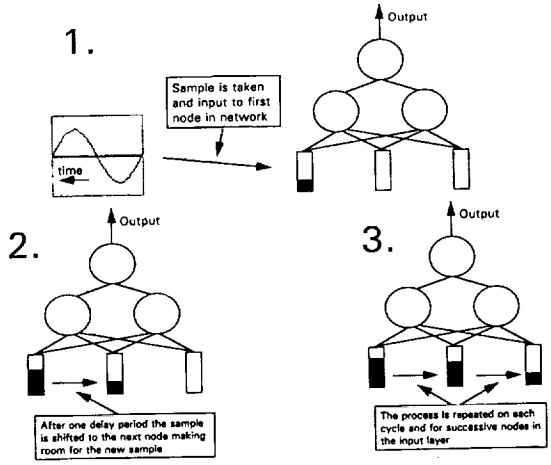


Figure 4. Sample, hold, and shift sample inputs to input layer nodes.

To monitor the temporal integrity of the input waveforms, other cooperative networks (Figure 5) can be constructed to keep track of the temporal ordering of data being put into and recognized by the signal recognition network. In the example case, the phase angle recognition network can be trained to output an analog value in the range (0,1) corresponding to the phase pattern number of the rightmost node value. Another network is then trained to detect the proper sequence of patterns coming out of the phase recognition network and provide an indicator signal whenever this order varies from the prescribed sequence. A fourth network combines the phase sequence indicator with the waveform recognition indicator to produce an overall indication of signal health.

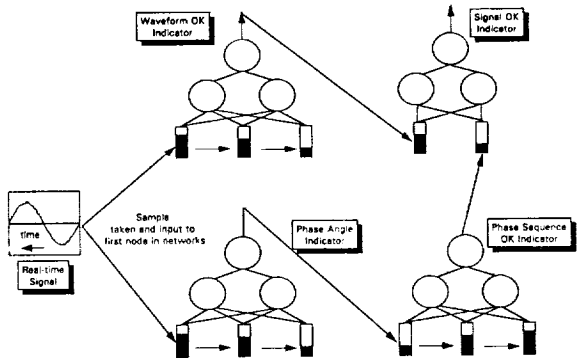


Figure 5. Cooperative signal recognition networks.

4.0 Telemetry Monitor Application

The application presented here is the monitoring of a simple sine wave by a cooperative network set while providing an indication of when it deviates from a nominal amplitude and frequency. This example also demonstrates the utility of the SSCG training method. When an [11,25,1] network was trained with 40 sine wave patterns, a modified backpropagation routine took 21440 function and 21440 gradient evaluations. The SSCG method required 20664 function evaluations and only 2040 gradient evaluations to reach the same level of convergence.

For this test application, it was desired to correctly identify a simple, continuous 1.0 Hz, 0.5 amplitude sine wave and to detect any deviations in amplitude or frequency. Any detected deviations were to be called to attention by the loss of a "good" signal indication.

The simulation takes advantage of the cooperative network concept by using four networks (as illustrated in Figure 5) to determine the input signal's "health." The first net is trained on the temporally windowed input signal waveform and outputs a "good/bad" (generally a one/zero output) signal depending on the waveform's degree of match with the internally represented, previously learned training set. A second net receives the same signal simultaneously and outputs a value corresponding to the phase angle of the input signal. This output value is temporally windowed into a third timing network which looks for a regular, repeating pattern of phase angles corresponding to a good waveform. The fourth and final network has two input nodes. One node receives the good/bad signal from the first signal recognition net and the second receives the good/bad signal from the phase timing network. This control network's single output node then indicates the interpreted state of the original input waveform.

This application made use of a PC-based program to train the four required nets and another program to link the four nets together cooperatively in a user-friendly (Windows 3.0) environment. Reasonable net sizing was determined by a short set of function evaluations studies.

Training sets for each of the four nets were defined as follows. For the signal recognition [11,25,1] net, a set of ten temporally related "bad" sine patterns of amplitude 0.45, ten bad patterns of amplitude 0.55, and 20 good patterns of amplitude 0.5 were constructed. For the phase recognition [11,15,1] net, ten 0.9 Hz bad patterns, ten 1.1 Hz bad patterns,

and 20 1.0 Hz good patterns comprised the training set. The [11,10,1] phase timing network training set consisted of five windows of bad random input values, ten windows of bad constant input values in the range from 0.0 to 0.9, and ten windows of properly phased timing values. The forth, [2,3,1] control network was trained with four patterns. Whenever its two input nodes received a good output from the signal and phase timing networks, it would output a good signal (1.0). When the signal network indicated trouble, the voting network was trained to output a bad value of 0.75 indicating failure. Phase timing failures would cause the network to output a bad value of 0.5. The failure of both recognition nets would cause a zero value to be output.

After training to an average system error of less than 1E-4, the weight sets from the four networks were merged together into a cooperative set and loaded into the Windows-based program.

The results of this demonstration are best illustrated by examining the output from the four networks during normal operation and after induced failures (Figures 6 and 7). After an initial startup transient, the signal recognition network outputs a good value of 1.0 for an input amplitude of 0.5. The periodicity of the output value, small though it is, may be the result of incomplete training. If this is so, it can possibly be minimized by using a larger training set and a smaller convergence tolerance. The phase recognition network shows the periodic ramping of the recognized phase angle of the signal, and the phase timing network outputs a good signal for this. The control network's concurring output stays at one for as long as the waveform is correct.

Two cases, where failures are introduced at the five second mark, demonstrate the ability of the network to detect small deviations ($\pm$0.05) in input signal amplitude. Figure 6a shows the signal recognition net's failure signal. Figure 6b shows what happens to the phase angle net's output. In this case, trained only with constant amplitude waveforms, the phase net mistracks the phase angle at the lower amplitude. Figure 6c shows the phase timing net's output, as it loses its timing at the lower amplitude. Note that it might be desired that for all cases of amplitude failures, a phase network would still output a good signal, as it would if it were truly measuring phase. Such a response would be made possible by adding waveforms of varying amplitudes to the training set. In this instance, the control net responded reasonably well (Figure 6d) with the correct output (0.75) for the high amplitude failure, but the low amplitude failure caused the net to oscillate between 0.0 and 0.75 as the phase timing net repeatedly lost lock.

A second test case was performed, this time with frequency failures of $\pm$0.01 Hz injected at the five second mark. The signal recognition network, trained with constant 1.0 Hz frequency patterns indicated failure in an oscillatory manner (Figure 7a). Likewise, the phase network and its cooperative timing network, whose output is plotted in Figures 7b-c, failed to track the failed signal's phase. The resulting oscillating control network's output in Figure 7d diligently tracked the phase timing and signal recognition output as best it could.

5.0 Future Applications

The work presented here demonstrates the feasibility of using artificial neural networks to monitor low frequency real-time processes. Additional studies reported elsewhere have demonstrated the ability to use this network architecture to actually control a single parameter dynamic system (e.g., spacecraft actuator spin rate).[11]

This new application of the SSCG algorithm makes it possible to train and examine several network architectures for a specific application in a relatively short period of time. This is of significant advantage until a more analytical approach to sizing

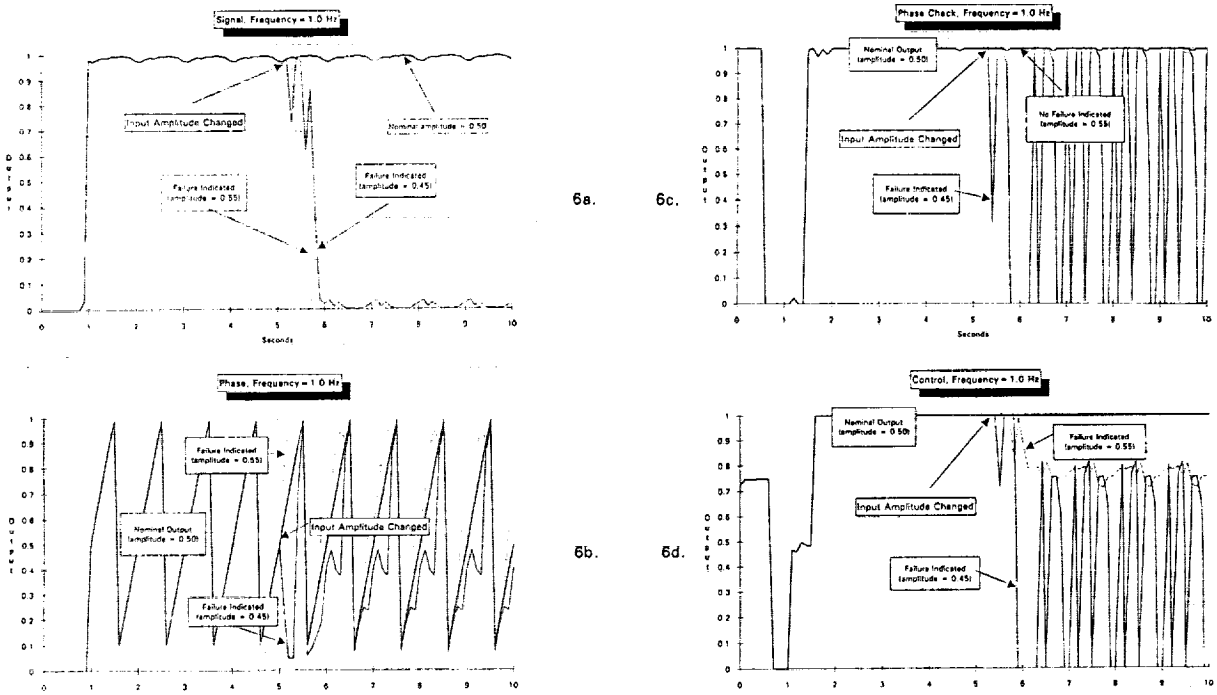


Figure 6. One Hz sine wave with induced amplitude faults.

neural networks becomes available. The availability of neural networks implemented in hardware will also speed up the training cycles and allow larger training sets to be presented in shorter periods of time.

Other topics related to telemetry monitor design are also open to further improvement and investigation. Real-time performance is a critical concern in the development of an intelligent space-borne configuration controller and telemetry monitor for controlling multiple subsystems. With large amounts of time-varying data, the validity and timeliness of conclusions based on instantaneous data is constantly in question. First attempts at using expert systems for real-time applications involved taking a snap-shot of data and using a static expert system to draw conclusions about system health. Conventional pattern-matching paradigms which examine all possible conclusions for the current data values are too slow for most real applications. Yet expert systems may still play a role in real-time controllers.

One approach would integrate a neural network front end with an expert system configuration controller. When performance exceptions are detected by the neural networks, an inference engine might invoke a set of metarules which would focus the attention of the inferencing system on the offending subsystem. The benefit of this approach is that knowledge bases with thousands of rules, properly gathered into smaller, related sets of rules, can be run in real-time.

Before neural network-based systems see everyday operations, the issues of verification and validation will also need to be addressed. When neural networks are called upon to generalize a desired response from an incomplete training set, it must be verified conclusively that the proper generalization was made. Accomplishing this within a finite amount of test time is a difficult issue yet to be fully studied.

Finally, future work in this area should also address the effects of input signal noise on the ability of the signal and phase recognition nets to discriminate their desired waveforms. Noise might also serve as a means to create a deadband or wider acceptance region around some nominal patterns used to train networks. Random amplitude noise, superimposed on top of the repeating, temporally related patterns can make those patterns appear to "cover more space" than the basic set. In this manner, a significantly large deadband effect might be learned by a network with only a small input pattern training set.

Artificial neural networks and their successors will soon find their way into the aerospace engineer's box of tools, much as the serial digital computer did over forty years ago. Their pattern recognition capabilities will complement the available tools, methodologies, and techniques in an untold myriad of ways. The SSCG training method presented here, along with cooperative neural network signal recognition concepts, represents yet another step in the exploration of the potential of using neural networks to monitor and control a variety of aerospace and other related systems.

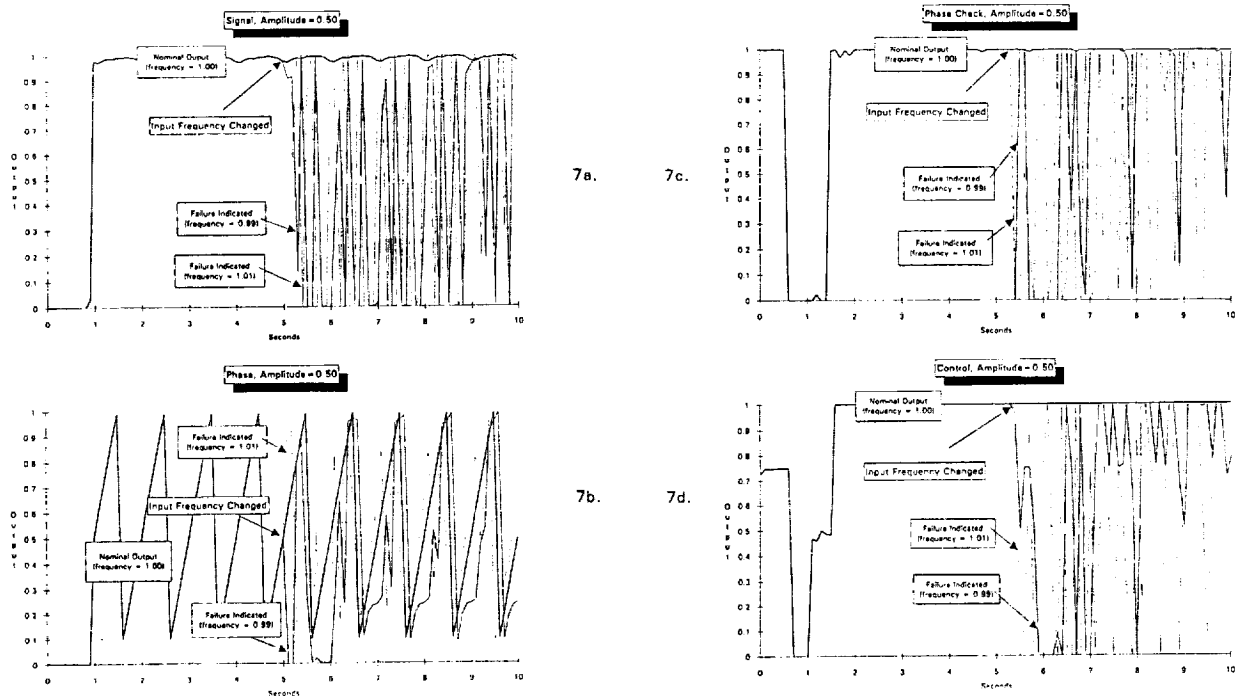
References

[1] Cybenko, G., "Approximations by Superpositions of a Sigmoidal Function," CSRD Report 856, UIUC Center for Supercomputing Research and Development, 1989, pp. 1-15.

[2] Doyle, R. J., Berleant, D. Falcone, L. P., and Fayyad, U. M., "Selective Simulation and Selective Sensor Interpretation in Monitoring," AIAA 89-3101-CP, 1989, pp 859-870.

[3] Fahlman, S., "Faster Learning Variations on Back-Propagation: An Empirical Study," Proc. of the 1988 Connectionist Models Summer School at CMU, Morgan Kaufman Publishing, 1988, pp. 38-51.

[4] Gargan Jr., R.A. and Kovarik Jr., V., "An Expert System for Mission Scheduling and Conflict Resolution," Proc. 2'nd Annual Workshop on Robotics and Expert Systems, Houston, TX, June 4-6, 1986.



7a. 7c.

7b. 7d.

Figure 7. One Hz sine wave with induced frequency faults.

[5] Gill, P., Murray, W. ,and Wright, M., Practical Optimization, Academic Press, 1981, p. 150, 99, 141, 146.
[6] Goldberg, K. and Pearlmutter, B., "Using Backpropagation with Temporal Windows to Learn the Dynamics of the CMU Direct-Drive Arm II," Advances in Neural Information Processing 1, Morgan Kaufmann, 1989, pp. 356-363.
[7] Groundwater, E.H., Lembeck, M.F., and Sarsfield, L., "Development of An Expert Planning System for the Office of Space Science and Applications," 4'th Annual Conference on Artificial Intelligence for Space Applications, Huntsville, AL, November 1988.
[8] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Vol 2, 1989, pp. 359-366.
[9] Kohonen, T., "An Introduction to Neural Computing," Neural Networks, Vol 1, 1988, pp. 3-16.
[10] Lambe, J., Moopenn, A., and Thakoor, A.P., "Electronic Neural Networks," NASA Technical Support Package, NPO-16680/6175, 1988.
[11] Lembeck, Michael F., Design Methodology for a Neural Network-based Telemetry Monitor. University of Illinois at Urbana-Champaign, Dissertation, March 1991.
[12] Lippmann, R., "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April, 1987, pp. 4-25.
[13] Lippmann, R., "Pattern Classification Using Neural Networks," IEEE Communications Magazine, November, 1989, pp. 47-64.
[14] Naidu, S. R., Zafirion, E., and McAvoy, T. J., "Use of Neural Networks for Sensor Failure Detection in a Control System," IEEE Control Systems Magazine, April, 1990, pp.49-55.
[15] Oren, S. S. and Spedicato, "Optimal Conditioning of Self-Scaling Variable Metric Algorithms," Math Programming, 10, 1976, p70-90.
[16] Pao, Y., Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading, MA, 1989.
[17] Rumelhart, D.E., McClelland, J.L., Parallel Distributed Processing, Vol 1, MIT Press, Cambridge, MA, 1986, p328, 330.
[18] Shanno, David F., "Conjugate Gradient Methods with Inexact Searches," Mathematics of Operations Research, Vol. 3, No. 3, August, 1978, pp244-256.
[19] Skapura, D. M., "A parallel Processor Designed for Artificial Neural Systems Simulation," AIAA 89-3090-CP, 1989, pp 798-803.
[20] Smith, R., "Structuring a Knowledge Base for Real-Time Diagnosis," Proc. of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space, Kobe, Japan, 1990, pp. 197-200.

N93-11934

FUZZY CONTROL SYSTEM FOR A REMOTE FOCUSING MICROSCOPE

Jonathan J. Weiss, Ph. D. and Luc P. Tran
McDonnell Douglas Space Systems Company
16055 Space Center Blvd.
Houston, TX 77062

ABSTRACT

Space Station Crew Health Care System procedures require the use of an on-board microscope whose slide images will be transmitted for analysis by ground-based microbiologists. Focusing of microscope slides is low on the list of crew priorities, so NASA is investigating the option of telerobotic focusing controlled by the microbiologist on the ground, using continuous video feedback. However, even at Space Station distances, the transmission time lag may disrupt the focusing process, severely limiting the number of slides that can be analyzed within a given bandwidth allocation. Substantial time could be saved if on-board automation could pre-focus each slide before transmission. The authors demonstrate the feasibility of on-board automatic focusing using a fuzzy logic rule-based system to bring the slide image into focus. The original prototype system was produced in under two months and at low cost.

Slide images are captured by a video camera, then digitized by gray-scale value. A software function calculates an index of "sharpness" based on gray-scale contrasts. The fuzzy logic rule-based system uses feedback to set the microscope's focusing control in an attempt to maximize sharpness.

The system as currently implemented performs satisfactorily in focusing a variety of slide types at magnification levels ranging from 10x to 1000x. Although feasibility has been demonstrated, the system's performance and usability could be improved substantially in four ways: by upgrading the quality and resolution of the video imaging system (including the use of full color); by empirically defining and calibrating the index of image sharpness; by letting the overall focusing strategy vary depending on user-specified parameters; and by fine-tuning the fuzzy rules, set definitions, and procedures used.

INTRODUCTION

The Space Station Crew Health Care System periodically collects microscope slides that must be analyzed by microbiologists on the ground within hours of collection. It would be desirable to minimize the time required for the crew to mount and focus microscope slides. However, simple telerobotic control by the microbiologist under visual feedback would entail several problems: first, a transmission delay of about two seconds in each direction will disrupt the focusing process, producing both slower and less satisfactory results than hands-on control would; second, because of the additional time required, the focusing process itself would tie up significant amounts of video downlink bandwidth, reducing the time available for the actual analysis; finally, the requirement to schedule slide mounting, focusing, image transmission, and analysis at the same time places tight constraints on all resources used, and may be highly inefficient and inflexible.

The system would be far more robust and efficient if the crew could collect the slide specimens and place the slides in a cassette or "jukebox", with no further crew attention required. An on-board robot could then mount each slide on the microscope, bring the slide into focus, and transmit one or more digitized still images to a computer on the ground. The microbiologists could then view and analyze the slides at their convenience. The resulting digitized images would require far less bandwidth to transmit than a continuous video image, yet each digital image could have better resolution and less noise than standard video.

Of course, in practice the microbiologist might still wish to be part of the focusing loop, either to concentrate on particular areas within a slide or to fine-tune focusing when the system's focus does not seem sharp enough. This could be accommodated by exception (recalling a slide that is still available in the "jukebox"), or

by allowing the microbiologist to preview each slide (in real time) and transmit a command that would modify or override the system's automatic focusing. Alternately (at greater cost but probably still less than full-video transmission), the system could transmit not one "focused" image but a series of images representing a range of focus settings.

The major technical challenge in this approach is the automatic focusing itself; mature technologies exist to handle the slide mounting, and to digitize, transmit, store, and display still slide images. This paper describes a demonstration prototype system that the authors developed at McDonnell Douglas Space Systems Company, with significant contributions by Apt Instruments, Incorporated (now Aptronix). The system demonstrates the feasibility of automatic focusing using fuzzy rule-based control and the possibility of producing similar control systems rapidly and inexpensively.

SYSTEM DEFINITION

Functions

Functionally, the system operates as a closed-loop discrete-time feedback system whose goal is to maximize the sharpness of a slide image by controlling the vertical position of the microscope stage (see Figure 1).

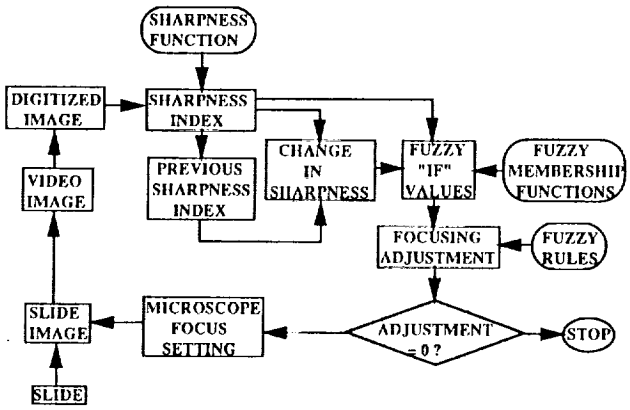


FIGURE 1

Slide images are captured by video and digitized. The system calculates a numerical scalar value representing the "sharpness" of any image from its digital record. This sharpness index and the difference between its current and previous values form the inputs to a system of fuzzy "if-then" rules that determines whether the maximum sharpness has been reached. If if has, the system stops and displays the resulting image. If the system has not reached its stopping point, the rule-based system specifies the amount and direction to adjust the microscope's focusing control. The result produces a new image, which starts the cycle again.

Hardware

Figure 2 shows the hardware used in the demonstration prototype system. This system was assembled to the authors' specifications by Apt Instruments, Inc., under contract to McDonnell Douglas Space Systems Company. An Olympus microscope with 10x, 100x, and 1000x magnification levels produces slide images that can be viewed directly, but are also transmitted to a Javelin black-and-white solid-state video camera. The video image is captured by a frame digitizer board on the 80386 personal computer. The digitized image can be displayed on a video monitor.

The 80386 personal computer performs the computations necessary to interact with the user, to compute the sharpness index, to run the fuzzy logic inference engine, and to specify desired focusing commands. These commands, in digital form, are transmitted to a motor controller, which translates them into pulses that drive a stepper motor. This motor, permanently mounted to the microscope's focusing mechanism, adjusts the vertical position of the microscope's stage.

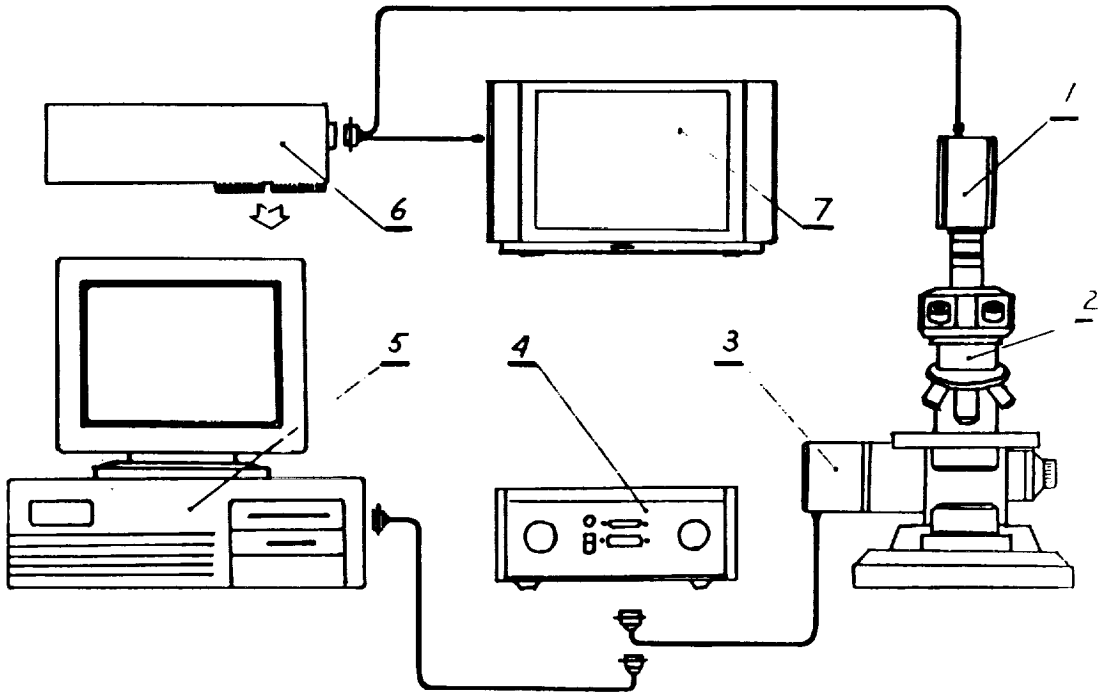
Software

Although much of the system's software covers routine functions such as system setup, input-output, etc., four components are worth describing in detail.

USER MENU

The user's interaction with the microscope system operates from a main menu of options. These are selected by pressing the function keys on the keyboard and are presented in logical sequence.

SET/RESET - This must be performed at system startup and whenever a new objective lens is selected. The user may also RESET the system at any time. The system prompts the user to select a level of magnification. (The current system requires the user to rotate the objective lenses manually; if the appropriate hardware were available it would take only minor software changes to control the lenses robotically.) Once a level of magnification has been selected, the user is prompted to set the absolute limits on the height of the microscope's slide stage. This prevents physical damage to the microscope or to the slide. (Again, a more sophisticated robotic system could perform this task automatically.)



SYSTEM HARDWARE CONFIGURATION

1. VIDEO CAMERA JE2362 2. MICROSCOPE BH-2 3. MOTOR M57-51 4. MACHINE CONTROLLER
5. IBM PERSONAL COMPUTER AT 6. FRAME GRABBER BOARD DT2855 7. MONITOR

FIGURE 2

FOCUS - FOCUS initiates the automatic focusing procedure, adjusting the height of the microscope stage until the sharpest image has been reached. When selected right after SET/RESET, FOCUS uses the entire digitized slide image as input to the computation of the sharpness index. When selected after AREA, it uses only the designated area(s).

MANUAL - In MANUAL operation mode lets the user control the microscope focus directly, using the computer keyboard. The "up" and "down" arrow keys move the stage up and down in fine gradations, while the "page up" and "page down" keys move in larger steps. After each keystroke, the stage is moved and the new image displayed.

AREA - This option permits the user to customize the focusing process by specifying up to three areas of the slide image. When one or more areas have been specified, portions of the picture outside such areas are disregarded when computing the sharpness index. This allows the user to exclude irrelevant objects, dust or scratches on the slide's cover slip, etc., which might otherwise interfere with the focusing calculations; it is also valuable at high magnifications, when the slide image may have objects that overlap at different depths or lie at an angle to the focal plane. Areas are selected by using the computer's mouse to draw an outline which is superimposed on the displayed digital image. Users also have the option to rate the "importance" (impact on the sharpness index) of the areas selected, choosing 'very high', 'high', or 'medium'. Once AREA has been completed, the next step is to select FOCUS again; because the sharpness index is computed differently, the FOCUS routine will in general select a different stage position as the best focus.

COLLECT DATA - The user is prompted to specify a (rectangular) area of the displayed image, using the computer's mouse. The selected area of the digital image is saved as a data file on the computer. This data could be used to provide a digital image that can be annotated or enhanced off-line, to document system activity or performance, or to provide input to image analysis programs.

RETRIEVE/DISPLAY IMAGE - The user can view a previously saved image file on the video display.

MODE - The user may select 'continuous' or

'single step'. 'Continuous' mode, which focuses the image without further action on the user's part, is the system's normal modeof operation. In 'single step' mode, the system stops after each focusing adjustment, giving the user time to view the image, to return to MANUAL mode, or possibly to save the image using the COLLECT DATA option.

COMPUTATION OF THE SHARPNESS INDEX

At the foundation of the focusing system is the idea that an image (represented as a rectangular array of digital gray-scale pixels) can be evaluated by computing a scalar-valued "sharpness index." This concept is appealing because once a suitable sharpness function has been defined, the problem of focusing can be reduced to finding the highest value for a function of a single variable (the height of the microscope stage) whose value is restricted to a known interval. Although there are a number of subtleties, technical difficulties, and empirical issues still to be addressed, the current demonstration prototype uses a fairly simple general-purpose sharpness function with generally successful results. (Some of these issues, with possible alternate approaches, are described later in this paper.)

The program calculates sharpness in the following manner:

(1) Reduce computation by sampling only a small number of lines from the image. Currently, the program samples 7 bands of three adjacent lines each.

(2) Treat each of the bands in the sample as a set of overlapping 3x3-pixel squares. For each 3x3 square, represent the gray-scale values as *a* through *i*, in the following configuration:

a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

(3) Compute horizontal contrast as $x_h = (c+2f+i) - (a+2d+g)$, and vertical contrast as $x_v = (a+2b+c) - (g+2h+i)$. Total squared contrast for each 3x3 square is computed as $x_{tot}^2 = x_h^2 + x_v^2$.

(4) Sum the total squared contrasts for all 3x3 squares in the sampled bands to arrive at the raw sharpness index.

Essentially, this formula is the sum of squared local gray-scale contrasts, computed on a sample of the total image. For scaling purposes, the program also computes a normalized sharpness index, assigning a value of 100 to the best-rated image that is scanned during the SET/RESET routine as the user specifies the highest and lowest possible stage positions. The rationale for the current method is that as a well-focused image is blurred, sharp local contrasts will turn into smoother gradients, thereby lowering the overall sum of squared local contrasts.

IMPLEMENTATION OF FOCUSING STRATEGY

The focusing strategy employed by the system is based on a set of fuzzy rules. At each step in the focusing process, the current value of the normalized sharpness index and the difference between the current and the previous values are inputs to a fuzzy inference process described below. The output of that process is a suggested magnitude and direction for adjusting the microscope. For certain values (corresponding to well-focused images that do not show significant improvement with further adjustment), the rule-based system will suggest a change of zero; at that point, the focusing process stops.

The rules themselves take the form "IF *Sharpness* IS <descriptor1> AND *Change_in_Sharpness* IS<descriptor2> THEN *Correction* SHOULD-BE <value>". The two descriptors are labels for fuzzy sets defined over the possible values for *Sharpness* or *Change_in_Sharpness*, respectively. The value is a signed number corresponding to the magnitude of the focusing adjustment suggested.

In the current system, the range of possible *Sharpness* values is mapped onto six overlapping fuzzy sets. These sets roughly correspond to linguistic labels such as "very poor", "poor", "medium", "moderately sharp", "sharp", and "very sharp". Each of these sets is simply a function that assigns an integer value from 0 to 127 to any *Sharpness* value. A set of membership functions is illustrated in Figure 3.

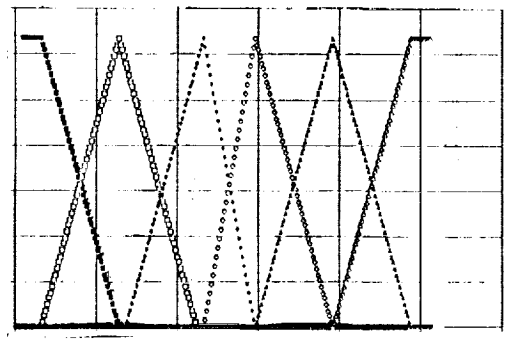


FIGURE 3

Similarly, *Change_in_Sharpness* has six membership functions. Thus, the system has a total of 36 rules.

Because the fuzzy values overlap, most of the time the rule input values (*Sharpness* and *Change_in_Sharpness*) match the "IF" conditions for two or more rules at levels greater than zero. When this occurs, the fuzzy inference technique used in this system computes a weight for each rule, which is the minimum of the two corresponding membership function values. All rules with nonzero weights are combined by taking a weighted average of the suggested outputs. For example, if Rule 1 had a weight of .25 and suggested an adjustment of +120 and Rule 2 had a weight of .75 and suggested an adjustment of +40, and all other rules had zero weight, the resulting suggested adjustment would be +60. This weighted-averaging technique assures smooth transition between adjacent sets of rules, an advantage compared to non-fuzzy rules which change actions abruptly as thresholds are crossed.

Special procedures are implemented to prevent undesirable situations. For example, if the image is not very sharp and does not appear to be improving, the rules may suggest a large step continuing in the same direction. But if such a step would exceed the limits defined in the SET/RESET process, the system adjusts the step to stay within limits. If this adjusted step still fails to improve, the system concludes that it has been looking in the wrong direction, returns to its starting point, and begins searching in the opposite direction.

SYSTEM PERFORMANCE

Operating on an 80386-based personal computer with an 80387 math coprocessor, the system generally reaches its "best focus" in a matter of seconds (corresponding to anywhere from 3 to about 25 focusing steps), as long as the slide contains sufficiently high contrast to generate meaningful values on the sharpness index. For very low-contrast slides, the system may fail to detect any objects and produce an error message during the SET/RESET procedure; slides that barely pass the initial test may cause the system to "search high and low" for any object, or oscillate between two settings without converging. It would be a simple matter to modify the code to detect this condition and terminate with an appropriate error message.

When the system reaches its "best focus", this means it has arrived at what it considers to be a sharp image. In some cases, this may differ from a human observer's judgment. Typically, it is possible to improve slightly on the sharpness of an automatically focused image by entering MANUAL mode and making fine adjustments. Part of this shortfall can be accounted for by the time-versus-quality tradeoff implicit in the focusing strategy and by the limited resolution of the microscope's motor; a more sensitive motor controller and a greater sensitivity to small changes in the sharpness value could increase the value of the sharpness function, but generally the magnitude of this improvement would be small.

The other component of the shortfall may be a mismatch between the sharpness index as computed by the system and the user's own concept of what constitutes a sharp image. Although preliminary investigations indicate that the current sharpness index performs adequately, more study is needed to determine if significant improvements can be obtained by defining the sharpness function differently. This may be particularly important at 1000x magnification, where greater shortfalls have been observed, possibly due to the more 3-dimensional nature of the 1000x images.

A side issue, arising during a pilot evaluation study with microbiologists as subjects, relates to the system's visual display quality. The standard-video, black-and-white images displayed on an ordinary video monitor were judged inadequate for practical used by microbiologists. Although full-color, high-resolution equipment would have solved this problem, it would have added significantly to the cost of the prototype demonstration system while shedding little additional light on the major technical issues addressed in the demonstration.

Even when the "best focus" agrees well with human observers' ratings, a second quality factor needs to be evaluated: the speed or efficiency with which the system reaches its focus. As already mentioned, the current system performs adequately, but occasionally it appears to make adjustments that a human observer would consider unusual. Improvements might be effected by modifying the rule set, fine-tuning the membership functions, or selecting an alternate method of reconciling overlapping rules.

To summarize, the current demonstration prototype has succeeded in its primary goal, which was to demonstrate the feasibility, low cost, and relative simplicity of a fuzzy logic approach to a complex control problem with space system applications. The system can focus most slides quickly and with adequate or better sharpness, although improvements on both factors should be possible.

TECHNICAL ISSUES

The purpose of the current prototype system

was to demonstrate the technical feasibility of fuzzy control for problems like microscope focusing. A working production version of such a system would require upgraded hardware and reprogramming (perhaps using a stand-alone microprocessor instead of a general-purpose computer). In addition to those changes, three technical issues need further study.

Defining the "sharpness" function

In the final analysis, even the most efficient system will fail to be effective if it is optimizing the wrong sharpness index. It is critical that the peak of the sharpness function correspond closely to the location human experts would select in manual mode. Beyond that constraint, it would be desirable if the sharpness function's shape were conducive to a fast and successful search.

The current method, a sum of squared local contrasts, works well for many images, but has some drawbacks. When there are two or more objects in different focal planes, for example, a larger object contains more pixels, and therefore contributes more to the sharpness index than a smaller one. Also, a "solid-colored" object exhibits local contrast only along its borders, while a striped, spotted, or other textured object shows contrast throughout its area and therefore contributes disproportionately to the sharpness index. Particularly at 1000x, some images also tend to produce optical artifacts (reflections or interference patterns) that appear as alternating bright and dark bands, generally around an object's edge; these bands are not real objects, but by contributing high contrast values to the sharpness index, they may throw the focusing process off. Finally, high-contrast extraneous objects such as dust particles, scratches on the slide, air bubbles, or water droplets tend to be located on the top and bottom of the slide and may "decoy" the focusing process away from the true targets.

The current method is only one of a vast family of possible "statistical" indices of sharpness. The computation of local contrast might use a 5x5 or larger region instead of a 3x3, or might weight the adjacent pixels differently. Instead of sum-of-squares, one could use the sum-of-absolute-values, the number or percent exceeding a given threshold, or the maximum contrast. Other statistical methods might use measues based on gray-scale or contrast-level entropy, statistical filters, or correlations.

These "statistical" sharpness measures share one drawback: they try to use the same index of sharpness for many different kinds of images in varying circumstances. By contrast, a human microbiologist knows what kind of slide is being mounted and has a pretty good idea of what to look for. For example, on one type of slide, a microbiologist may be interested only in a particular band of tissue, and in particular may need to know whether the band is a single unit or a set of adjacent components arranged end-to-end. If the crew member who collected each slide could simply indicate which type of slide it was, a sophisticated system could use expert rules to select the most appropriate statistical measure for each one. (A further extreme of sophistication might use advanced techniques in artificial intelligence, fuzzy logic, or neural nets to categorize the slide automatically and zero in on features of high interest; this would, however, go well beyond the original scope of this project.)

Strategies and tradeoffs for "optimization"

In defining an "optimal" search strategy, we implicitly make trade-off judgments about the system's desired performance. Once a sharpness measure has been chosen, each slide image could potentially generate a sharpness-versus-stage-position function defined over the range of possible stage positions. The problem is to arrive "sufficiently close" to the maximum value in an "acceptably short" time, with "high reliability". Although the system designer must work from an educated guess, the final evaluation of a system's performance must address all three issues, and must agree with the user community's subjective perceptions.

In the case of microscope focusing, there are three conditions that may complicate the focusing strategy:

(1) *Multiple local maxima.* Since it is not known at the outset what the maximum achievable sharpness index is, there is no guarantee that any local maximum is in fact the global maximum. However, if the sharpness function is reasonably well-behaved (free of tall spikes), it should suffice to start by taking coarse steps, then progress to finer adjustments.

(2) *Plateaus.* Sometimes, the sharpness index is fair to poor, but small to moderate-sized adjustments do not yield significant changes. This occurs often when the image is low-contrast, improperly lit, or far out-of-focus. The current system's approach is to continue in one direction until the sharpness index changes or until the upper or lower limit is reached; often this strikes the user as inefficient "blind searching". One possible approach might be to use a different kind of sharpness function until a certain quality of focus is reached, then transfer over to the "real" sharpness function. Along with this,

the rules might be modified (or an auxiliary set of rules developed) to take larger steps when on a plateau.

(3) "High-frequency" fluctuations. Particularly in the neighborhood of the maximum sharpness, very small adjustments seem to produce fairly large jumps in the calculated sharpness index. Some of this may be due to the resolution of the image digitizer, some to variations in the light source, and some to uncontrolled motion of the microscope or slide. Even at its finest adjustment, the stepper motor does not return the stage to its exact original position after one step up and one step back down. If these variations reflect the system's noise level, it would be advisable to modify the search strategy (or the sharpness function) to keep the system from getting caught in a prolonged series of very minor adjustments. If they represent the sharpness contributions of several values at distinct but close focal distances, there may be no universally satisfactory way to choose among them; the best solution may be to get the microbiologist into the decision loop, or to transmit several images instead of one. As in the case of the plateau, there might need to be a different set of rules for the case where sharpness is very good.

Fuzzy rule implementation

The current system uses one of several possible interpretations of fuzzy rules. In essence, it computes the degree to which the input data match the "IF" pattern for each rule and uses it as a weight for the numerical value specified by the "THEN" portion of the rule, taking a weighted average to reconcile when multiple conflicting rules "fire". This approach, which was developed by Apt Instruments, Inc., is conceptually simple and computationally efficient, but there is no evidence to date that compares its performance with other possible techniques. Therefore, it might be both instructive and prudent to compare several techniques to determine relative performance.

Another area for improvement would incorporate some sort of feedback or adaptive learning into what is now a static set of fuzzy rules and membership functions. This would most likely begin with an intuitive fuzzy model based on an expert's behavior or verbal instructions and, then, modify the rules and membership functions based on experience. One possible use might be to begin with a generic set of fuzzy rules and then develop a specialized version for each different type of slide.

CONCLUSION

The authors have demonstrated the feasibility of using a fuzzy rule-based system to control the automatic focusing of a microscope. Although several enhancements have been suggested, the current demonstration prototype illustrates that a very simple rule base and inference engine can be used to guide the focusing process successfully. The ability to produce working prototypes quickly and at low cost, coupled with the possibility of capturing the control process on small stand-alone processors, suggests that fuzzy rule-based systems may be an attractive way to implement automation in space.

ACKNOWLEDGMENTS

The demonstration prototype system was produced as part of a McDonnell Douglas Space Systems Company (MDSSC) IRAD project (PD-503). Initial system specifications were produced by Dr. Jonathan Weiss, Dr. Jack Aldridge, and Luc Tran, all of MDSSC. Dr. Hongmin Zhang, Dr. Xiwen Ma, and Weidong Xu, all of Apt Instruments, Inc. (now Aptronix) assembled the system hardware, developed the code to control the system's hardware interfaces, and supplied an initial version of the fuzzy logic code, under contract to MDSSC. Subsequent modifications to the system were made by Luc Tran and Jonathan Weiss.

N93-11935

FUZZY LOGIC CONTROL FOR CAMERA TRACKING SYSTEM

Robert N. Lea
Johnson Space Center

R.H.Fritz
Link Flight Simulation

J. Giarratano
University of Houston Clear Lake

Yashvant Jani
LinCom Corporation

ABSTRACT :

A concept utilizing fuzzy theory has been developed for a camera tracking system to provide support for proximity operations and traffic management around the Space Station Freedom. Fuzzy sets and fuzzy logic based reasoning are used in a control system which utilizes images from a camera and generates required pan and tilt commands to track and maintain a moving target in the camera's field of view. This control system can be implemented on a fuzzy chip to provide an intelligent sensor for autonomous operations. Capabilities of the control system can be expanded to include approach, handover to other sensors, caution and warning messages.

1. INTRODUCTION

Advanced sensor systems with intelligence and a distributed nature will be required for activities such as proximity operations and traffic control around the Space Station Freedom (SSF). These systems will receive various types of measurements from multiple sensors and perform the necessary data fusion for the Navigation, Guidance and Control systems. The SSF operational requirements necessitate that this system be composed of passive type low power sensors.(Based on the current design, the SSF operations are expected to be power limited and computing resources limited.) An important feature is that the system should be capable of handling imprecise and approximate measurements as well as sensor failures.

A number of theories dealing with approximate reasoning such as fuzzy logic and Dempster-Shafer theory, were considered for this work and fuzzy logic was selected. Since its inception by Lotfi Zadeh [1, 2] in the 1960's, fuzzy logic has been applied to many fields [3, 4, 5], to handle imprecise measurements. Applications of fuzzy logic have also been developed for the star-tracking system of the Space Shuttle [6], the attitude control [7] and a combined translational and rotational control of a spacecraft [8].

The concept development for such an advanced sensor system that can accept measurements from several cameras and laser rangefinders is in progress within the Software Technology Laboratory of the Information Systems Division at the Johnson Space Center. The first phase of this development is a Camera Tracking System based on a fuzzy logic approach that utilizes the object's pixel position and range as inputs and controls the camera gimbal drives to keep this object in the Field Of View (FOV) of the camera. Later phases will involve development of other functions as described in the future activity section.

In this paper, we describe our concept of a fuzzy logic based tracking controller that meets these requirements. A typical proximity operations scenario and the tasks of a tracking system are discussed in section 2. The fuzzy logic approach to the tracking controller and details of the membership functions and rulebase are described in section 3. Advantages of the fuzzy logic approach for such a system are discussed in section 4. Future activities and a summary are given in section 5.

2. OPERATIONS SCENARIO

The proximity operations zone around the SSF is defined as a sphere of about 2000 feet. Within this zone, several activities [9] such as fly-around, final approach and docking will be performed involving other vehicles such as a Space

Shuttle, satellite servicer and other payloads. The zone also includes extravehicular activities performed by the crew in spacesuits or by telerobots to inspect the SSF structure and subsystems and, if necessary, install and/or replace the components. Some of the payloads approaching the SSF will require extensive involvement by the crew. All extra-vehicular activities, trajectories of incoming vehicles and all docking and berthing operations must be closely monitored to avoid collisions.

A typical scenario for proximity and docking operations with the SSF is shown in fig. 1. Assume there is an autonomous satellite approaching the station from about 2000 feet. It first moves to about 200 feet, then performs station keeping at that point for a short period of time. It then completes the necessary information exchange with the SSF, and proceeds to a closer point in such a way that the mobile remote manipulator system moving on the SSF structure can grapple it. This autonomous satellite, if necessary, can also perform a 'fly-around' to observe the SSF from various angles. During all these activities, cameras mounted on the SSF monitor the motion of the satellite and advise the crew in case of malfunction or irregularities. High power sensors like radar and lasers are less desirable when operations are power limited.

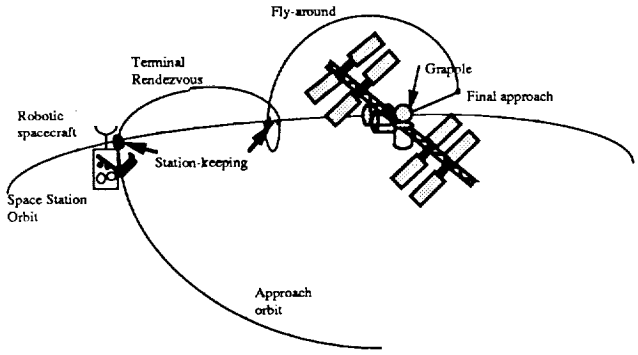


FIG. 1 OPERATIONS SCENARIO FOR TRACKING TASKS

Tracking of an object means aligning the pointing axis of a camera along the object's line-of-sight. vector The monitoring camera is typically mounted on the pan and tilt gimble drives which are capable of rotating the pointing axis within a certain range. The task of the tracking controller is to command these gimble drives so that the pointing axis of the camera is along the line-of-sight vector which is estimated from the measurements.

There are several issues the tracking controller must handle properly while commanding the gimble drives. Since the objects could be fast or slow moving, the controller must keep track of the rotating speed properly. Furthermore, the image could be blurry with a low resolution so that the estimate of the line-of-sight vector is poor. Handover to another camera tracking system may be required because of physical limitations of gimble drives and its mounting geometry. The controller must be aware of its gimble limits and whether the drives are approaching these limits.

There are two methods of commanding these gimble drives: 1) Rotate the gimble by the desired delta angle, and 2) Achieve the desired angular rate for that axis. In the tracking system under development, the gimble drives will be commanded using the second method. The object's line-of-sight vector will be estimated from the position measurements in terms of pixels (as shown in fig. 2) and is input to the control system. There is a laser rangefinder that provides a range measurement for the object in the FOV. This information can be used to properly estimate the effects of the objects speed on the rotation of the line-of-sight vector.

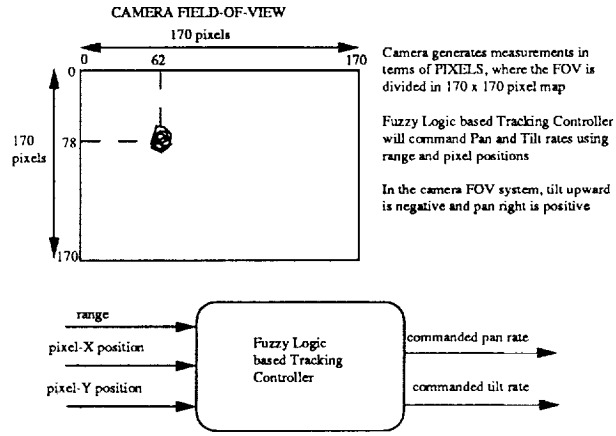


Fig. 2 CONCEPT OF A CAMERA TRACKING SYSTEM

3. FUZZY LOGIC APPROACH TO CONTROL

For the fuzzy logic based tracking controller, the inputs are range and line-of-sight vector, and the outputs are the commanded pan and tilt rates. The line-of-sight vector is input in terms of pixel position in the camera FOV. When an image is received, it is processed to determine the location

of the object in the camera frame which has the vertical, horizontal and pointing vectors as three axes. Usually an image, particularly for complex objects, spans over many pixels. Using a suitable technique, the centroid of the image is computed as the current location in the viewing plane which is like a Cartesian coordinate plane having vertical and horizontal axes. The size of the viewing plane is 170 x 170 pixels, and the origin is at the upper left corner as shown in fig. 2. The range of the object is received from the laser rangefinder as a measurement. These three parameter values are input to the controller.

Each of these parameters have their respective membership functions as shown in fig. 3. The range membership functions are Very_Far (VFAR), FAR, NEAR, Very_Near (VNEAR) and Proximity (PROX). The Proximity membership function represents a close proximity threshold within which docking operations can begin. The universe of discourse for range is from 0 to 200 feet. If the spacecraft operations begin at 2000 feet, then this universe of discourse can be extended to that value, however, a large part is then contained in only one membership function, VFAR.

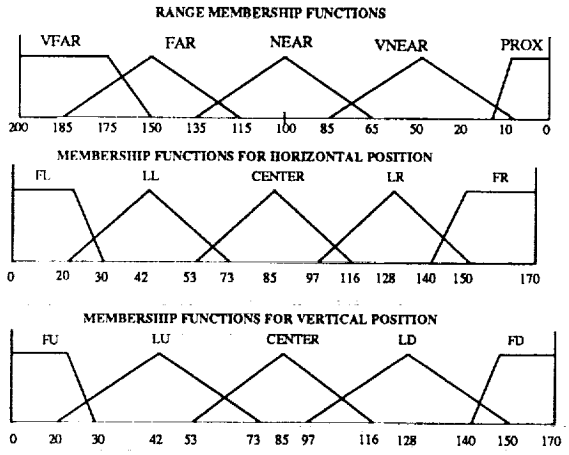


Fig. 3 Membership functions for the input parameters

The horizontal position membership functions are Far_Left (FL), Little_Left (LL), CENTER, Little_Right (LR), and Far_Right (FR). The vertical position membership functions are Far_Up (FU), Little_Up (LU), CENTER, Little_Down (LD), and Far_Down (FD). The universe of discourse for the horizontal and vertical pixel positions is from 0 to 170. The typical FOV is about 45 degrees wide, however, the range of a gimble drive can be from -180 to 180 degrees and thus it should be noted that the FOV range does not represent the gimble drive range.

The membership functions shown in fig. 4 for the pan and tilt rates are the same with a universe of discourse from -6.0 to 6.0 degrees per second implying that the maximum rate that can be achieved by gimble drives is 6.0 degrees per second in one direction. Membership functions are Fast Negative (FN), Slow Negative (SN), Zero (ZR), Slow Positive (SP) and Fast Positive (FP). To properly handle the different ranges and effects of the velocity of objects in the FOV, there is a need to generate a scale factor. If the objects range is very far, then, with a typical approach velocity, its line-of-sight vector is not going to rotate fast. Thus, the pointing vector rotation can be managed with a low rate. If the object is very near or in the proximity range, then the approach velocity may rotate the line-of-sight vector with a high rate. In that case, the object may go out of the FOV quickly. In order to keep up with high angular rate of line-of-sight vector, the controller will need a high rate. The scale factor membership functions, shown in fig. 4, provide the capability to change the response based on the range measurements. All membership functions are triangular shaped for simplicity in implementation.

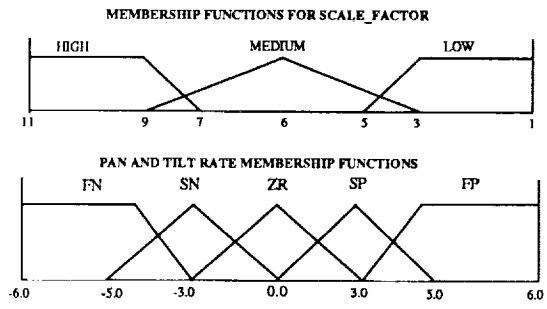


Fig. 4 Membership functions for the output parameters

The desired image location is the center of the viewing plane, which is at (85,85). If the current location is close to the center, then rotation of the pointing axis is not required. If the location is to the left of center then a left rotation is necessary. Similarly, if the image is down from the horizontal line then a downward rotation is required. These rotations are determined using the

position and range measurements and the rulebase shown in Table I. First the range measurement is fuzzified and the value of the scale factor is determined based on the scale_factor rules. The necessary defuzzification processing is performed to compute the crisp value of the scale factor. Then, the crisp scale factor and the position measurements are provided to the next set of rules to determine the rate at which the gimble drives should be rotated. There are 30 rules that determine both pan and tilt rates. Again, the necessary defuzzification processing is performed to compute the crisp values of the pan and tilt rates which can be sent to the gimble drives as command values.

Table I. Rule base for the tracking task

	Distance Membership Functions				
	VFAR	FAR	NEAR	VNEAR	PROX
Scale_Factor	LOW	LOW	MED	HIGH	HIGH

Scale_Factor	Horizontal Position Membership Functions				
	FL	LL	CENTER	LR	FR
LOW	FN	SN	ZR	SP	FP
MED	SN	SN	ZR	SP	SP
HIGH	SN	ZR	ZR	ZR	SP
	Pan_Rate Membership Functions				

Scale_Factor	Vertical Position Membership Functions				
	FD	LD	CENTER	LU	FU
LOW	FP	SP	ZR	SN	FN
MED	SP	SP	ZR	SN	SN
HIGH	SP	ZR	ZR	ZR	SN
	Tilt_Rate Membership Functions				

Note - Negative Tilt_rate means the pointing axis going upward in FOV

KEY : VFAR - Very Far, VNEAR - Very Near, PROX - Proxomity zone
FL - Far Left, LL - Little Left, LR - Little Right, FR - Far Right
FU - Far Up, LU - Little Up, LD - Little Down, FD - Far Down
FN - Fast Negative, SN - Slow Negative, ZR - Zero,
FP - Fast Positive, SP - Slow Positive

The camera is moved based on these commands within the limits of its gimble rates and angles. New measurements in the camera FOV are obtained for the next cycle and the processing is repeated. The cycle time is based on the time requirements for the following functions: 1) determine pixel positions, 2) obtain a range measurement, 3) rotate the gimble drives at a desired rate, and 4) the requirements to track the object within a certain performance envelope. Typical cycle time ranges between 0.1 to 1.0 second.

4. ADVANTAGES OF FUZZY LOGIC APPROACH

There are several advantages of our approach that utilizes fuzzy logic in a camera tracking system. The fuzzy logic approach is simple to understand and easy to implement as a software module. Fuzzy rules provide a framework to implement the human thinking process i.e. the rules reflect the human thought process, such as " If the object is Far_Left then rotate the camera to the left side ". The entire rule base is derived as if a human was performing the tracking task.

Implementation of fuzzy membership functions, rules and related processing is made easy by tools like the TIL Shell [10] which has a graphics oriented user interface and fuzzy-C compilers [11] that can generate code for the fuzzy chip or the C code to integrate with other software modules. There are several commercial products available in the industry.

It is also possible to develop and implement the fuzzy controller in the 'fuzzy processors', thus, having a fuzzy hardware controller. There are several commercial fuzzy processors [12, 13, 14] that can process over 30,000 fuzzy rules per second and thus provide a high speed processing power. These fuzzy processors consume low power with a capability to process general purpose instructions and can be mounted in the back plane of a camera. These processors also provide interfaces to hardware to transfer information and commands. Advanced sensor system envisioned for space station operations will have such processors embedded as an integral part of the system.

The camera tracking system described here can be built as an intelligent sensor with built-in intelligence and speed to perform functions which are normally performed in the distributed processing network of approximately 20 computers (the actual number of computers may change with requirements) onboard SSF. Because of a dedicated fuzzy chip and its processing power, there is virtually no computational load to the SSF

computing network. As a result, the SSF computers will be available for other computing requirements such as complex guidance and navigation schemes. Furthermore, the interfaces between the fuzzy chip and the computing network will be at a command level requiring reasonably low speed data transfer. There will be no need for a high rate data transfer which can increase the cost and decrease the reliability.

This system will involve low power sensors as compared to an active sensor e.g. Radar in Ku band range, or LADAR using laser frequency. Typically, the active sensor radiates a power pulse towards a target and receives back a reflected pulse. Based on the power transmitted, power received and time between these pulses, parameters like range and range rates are calculated. Since the camera tracking system will not be radiating power, it will be a low power system in comparison with active sensor system. Since there is already a shortage of power, an important consumable, onboard the SSF, availability of low power sensors is very important for continuous operations. The SSF can afford to keep this type of a sensor working around the clock without having much impact on the power management or other computational load on the main computers.

Capabilities of the tracking controller can be expanded to perform other functions such as approach toward the object, grapple, object identification, traffic management, and caution and warning to the crew. Fast moving objects can be identified easily via prediction of position and thus collision avoidance can also be achieved. Since the system can work as a stand-alone system at the command level and will interrupt the operations flow only if necessary, it can become a node in a distributed intelligent sensor system.

5. FUTURE ACTIVITIES AND SUMMARY

Our future activities include testing of the concept in software and hardware simulations. The software testing will help fine tune the rule base and the membership functions, while the hardware testing will help to identify and solve integration issues. Both types of testing are required in order to make the system operational and useful. All interface problems need to be resolved to implement the controller on a fuzzy hardware chip. Performance of the controller must be evaluated in light of real-time response, accuracy and imprecise measurements.

In later phases of the development, it is planned to expand the concept and fuzzy logic approach from tracking to : 1) identifying potential sensor failures and notifying the crew 2) handover techniques for switching to a different type of sensor measurement, 3) identifying one or more objects appearing in the camera's FOV (object extraction and identification algorithms), and 4) traffic management guidance. For configurations where the camera is mounted on a movable platform or on an end-effector of a robotic arm, this concept will be expanded to perform tasks such as the approach towards the docking fixture of the object, and grappling and rigidizing operations (similar to the operations performed by the crew). The concept can also be expanded to incorporate sensor data fusion required for the debris avoidance task.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Dr. Sankar Pal, Dr. Mike Murphy and James Villarreal for helpful and interesting discussions, and Jeffrey Hoblit, Hana Shehadeh and Indranil Chowdhury for the programming and documentation support.

REFERENCES :

1. Zadeh, L. A. : "Fuzzy Sets", Information and Control, vol. 8, pp. 338-353, 1965.

2. Zadeh, L. A. : "Outline of a new approach to the analysis of Complex Systems and Decision Processes", IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-3, no. 1, June 1973.

3. Klir, G. J. ; and Folger, T. A. : Fuzzy sets, Uncertainty, and Information, Prentice Hall, Englewood Cliffs, N. J., 1988.

4. Zimmermann, H. J. : Fuzzy Set Theory and Its Applications, Kluwar-Nijhoff Publishing, Boston, MA.,1985.

5. Giarratano, J. ; and Riley, G. : Expert Systems Principles and Programming, PWS Kent, Boston, MA., 1989.

6. Lea, R. N. ; and Giarratano, J. : "An Expert System Program Using Fuzzy Logic For Shuttle Rendezvous Sensor Control", Proceeding of ROBEXS'86, pp. 327-329, 1986.

7. Lea, R. N. ; and Jani, Y. : "Spacecraft Attitude Control System Based on Fuzzy Logic Principles", Proceedings of ROBEXS'89, 1989.

8. Lea, R. N. ; Togai, M. ; Teichrow, J. ; and Jani, Y. : "Fuzzy Logic Approach to Combined Translational and Rotational Control of a Spacecraft in Proximity of the Space Station", Proceedings of the IFSA'89, pp. 23-29, 1989.

9. Sedej, D. T.; and Clarke, S. F. : Rendezvous/Proximity Operations Workbook, Mission Operations Directorate, Johnson Space Center, Feb. 1985.

10. Hill, G.; Horstkotte, E.; and Teichrow, J. : TIL Shell User's manual, v2.0b, Togai InfraLogic Inc. Irvine, California, April 1989.

11. Teichrow, J. ; and Horstkotte, E. : Fuzzy-C Compiler User's Manual, Togai InfraLogic Inc., Irvine, California (1989)

12. Togai, M. ; and Corder R. J. : "A High Speed Fuzzy Processor for Embedded Real-time Applications", Proceedings of SICE, 1989.

13. Corder, R. J. : "A High Speed Fuzzy Processor", Proceedings of IFSA89, pp. 379-381, 1989.

14. Watanabe, H. : "VLSI Chip for Fuzzy Logic Inference", Proceedings of the 3rd International Fuzzy Systems Applications Congress, pp. 292-295, 1989.

INTELLIGENT DATA MANAGEMENT FOR REAL-TIME SPACECRAFT MONITORING

Ursula M. Schwuttke

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr., Pasadena, CA 91109
ursula@nereid.jpl.nasa.gov

Les Gasser and Bruce Abramson

Computer Science Department
University of Southern California
Los Angeles, CA 90089
gasser@usc.edu

Abstract

Real-time AI systems have begun to address the challenge of restructuring problem solving to meet real-time constraints by making key trade-offs that pursue less than optimal strategies with minimal impact on system goals. Several approaches for adapting to dynamic changes in system operating conditions are known. However, simultaneously adapting system decision criteria in a principled way has been difficult. Towards this end, a general technique for dynamically making such trade-offs using a combination of decision theory and domain knowledge has been developed. The paper discusses multi-attribute utility theory (MAUT), a decision theoretic approach for making one-time decisions, describes dynamic trade-off evaluation as a knowledge-based extension of MAUT that is suitable for highly dynamic real-time environments, and provides an example of dynamic trade-off evaluation applied to a specific data management trade-off in a real-world spacecraft monitoring application.

1. Introduction

Lengthy response times often prohibit optimal problem solving in the presence of real-time constraints. Effective real-time systems therefore require meta-reasoning for making appropriate trade-offs and, when necessary, pursuing less optimal methods. Such meta-reasoning often must take place in the presence of incomplete information, insufficient resources, and unpredictable situations; precise mathematical approaches with parallels in traditional control theory therefore cannot be formulated. As a result, the applicability of decision theory and the psychology of judgment to this problem area was recognized early, with research on heuristic methods for inference control [Simon 1955]. However, initial enthusiasm for using decision theory as an artificial intelligence technique dwindled in favor of approaches that seemed to lend themselves more naturally to expressing the rich structure of human knowledge [Horvitz 1988].

Uncertainty in reasoning has since been expressed with probabilities and statistics and has been thoroughly explored for nonreal-time AI applications in the context of Bayesian belief representation [Pearl 1988], [Shachter 1987]. However, degrees of uncertainty in real-time situations can change rapidly, imposing overwhelming complexities on these techniques. Bayesian statistics relies on the availability of conditional probabilities for the various hypotheses and pieces of evidence that pertain to a given situation. A common implementation of Bayesian statistics appears in medical expert systems that calculate the probability that a patient is suffering from a particular disease, given a manifested set of symptoms. Even in this relatively simple application with slow trend changes, the prospect of deriving the needed statistics is not straightforward: it is beset by a multitude of questions and variables such as when to use global statistics rather than local ones, how often to update models to reflect changing trends, and, more fundamentally, how to get access to valid information given inaccuracies and varying procedures in disease reporting. A central difficulty associated with the use of Bayesian statistics is therefore centered around dependence on stable information about the domain environment: this information, which is difficult to obtain even in simpler real-world situations, can be impossible to derive dynamically for complex real-time problems.

For such reasons, there has been renewed interest in decision theory for real-time AI applications. Rapidly changing circumstances require making trade-offs and expressing judgments, two processes which can entail a substantial level of subjectivity [von Winterfeldt 1986] and are therefore incompatible with rigid methods of analysis that require stable and accurate information. Decision theory provides a key ingredient: flexibility. This flexibility is embodied in formal decision-theoretic principles for obtaining preferred courses of action in the presence of uncertain events and conflicting objectives.

The simultaneous consideration of time pressure, complex environments, and potentially conflicting objectives has been studied in several different settings, including game playing [Russell 1989] and medical decision-making [Hor-

vitz 1989]. Studies of rational agents with time constraints have also begun to emerge in the literature [D'Ambrosio 1990], [Hansson 1990]. Most of these studies, however, rely upon assumptions that are not universally applicable. Both game-playing and medical diagnosis, for example, are self-contained domains with relatively long time lapses allowed between stimulus and response. Many interesting domains, such as spacecraft monitoring, violate both assumptions: the number of potential variables is huge and response time is negligible. The problem of automated spacecraft monitoring provides a perfect example of a real-time environment that is characterized by changing circumstances, uncertain events, and conflicting computational objectives. Several real-time systems with knowledge-based components have been developed for this complex application domain, [Laffey 1988], [Muratore 1990], [Schwuttke 1990] but these systems have focused primarily on being fast enough to handle expected computational loads and not on responding dynamically to unforeseen changes in real-time demands.

2. Dynamic Trade-Off Evaluation

Multi-attribute utility theory offers a natural way for dealing with competing objectives and is computationally straightforward, but has not been applied in dynamic real-time environments. Although a variety of static techniques from multi-attribute utility theory exist, only three variants of these techniques have been commonly applied to real-world situations [von Winterfeldt 1986]: the simple, multi-attribute rating technique [Edwards 1977], difference value measurement [Dyer 1979], and subjectively expected utility (SEU) measurement [Keeney 1976]. These approaches consist of the same general procedures and have collectively become known as SEU techniques. Edwards' technique is not only the simplest computationally, but also the most amenable to combination with knowledge-based approaches. It has thus been selected for our extension to dynamic real-time environments.

Our approach to this extension is to modify the basic SEU procedures while attempting to maintain their inherent simplicity, robustness, and flexibility. Our procedure is termed Dynamic Tradeoff Evaluation (DTE). In DTE, utility theory is used to rank alternatives in a preference space, and knowledge-based decision rules are used at run-time to 1) dynamically re-weight the attributes of individual alternatives and 2) to dynamically select among preference criteria in the preference space (depending on situational attributes and operational mode). The DTE methods are sufficiently general that they are applicable to a variety of run-time trade-offs, and are currently being applied to several very different real-time, real-world trade-offs in the domain of spacecraft monitoring. (See [Schwuttke 1991], which discusses and classifies a large range of potential trade-offs in Real-time AI.) The DTE methods are sufficiently general that they are applicable to a variety of run-time trade-offs and to integration into a real-time monitoring architecture.

The DTE procedure involves a sequence of six steps, many of which are derived from the steps of static SEU procedure. The first three of these steps and part of the fourth must be completed during the design phase of the system. For a given trade-off, the procedure includes:

1. *Definition of the trade-off instantiation mechanism.* This step involves specifying the circumstances under which DTE is required and designing the mechanism that will detect those circumstances and invoke the trade-off evaluation.

2. *Definition of application-specific alternatives and criteria that determine the value of the alternatives.* During this step, the alternative actions to be considered in the trade-off evaluation are specified, along with criteria that will be used to evaluate the alternatives. As part of this process, the system designers and domain experts also specify domain knowledge and (if necessary) heuristics that define the various ways of implementing each alternative. In addition, the decision criteria that influence the specific implementation of a run-time alternative are considered.

3. *Separate evaluation of each alternative.* This is done in conjunction with the previous step, and involves reliance on subjective judgements in cases where no basis for objective evaluation exists. Each alternative is ranked with respect to each of the evaluation criteria, on a scale of 0 to 100, and suitable consistency checks are applied to the evaluation.

4. *Definition of weights and modes.* Relative weights are assigned to each of the criteria, along with ranges within which the weights can vary. Domain knowledge is specified to determine the circumstances under which the weights will be varied. In addition, multiple modes may be specified, where each mode is governed by a different set of weights. Both the variation of the weights and the choice of a mode are determined at run-time using domain knowledge. These decisions are based on instantaneous circumstances in the monitored environment.

5. *Aggregation.* The weights selected in the previous step are used to determine the aggregate value of each of the alternatives, using the additive aggregation model put forth in SEU. These aggregate values provide the evaluation of the alternatives with regard to one of the trade-off axes. Depending on the specific trade-off, similar evaluation and aggregation may be required with regard to the second trade-off axis. However, in many cases the evaluation on the second axis may be directly calculated based on the dynamics of the environment. (In applications that do not require domain knowledge, the evaluations on both axes may be directly obtainable, but these applications are considered peripheral to this research).

6. *Selection.* An alternative is selected based on greatest total value with respect to both trade-off axes, as specified in the SEU methods. When the evaluation indicates that two or more alternatives are equally good, domain knowledge is used to select one alternative over the others, or if the alternatives are not mutually exclusive, to select several of them.

3. Application: Telemetry Data Management

We describe the application of DTE by reference to its application in a real-world spacecraft monitoring problem: managing input data for real-time knowledge-based monitoring of telemetry data from the Galileo Solid-State Imaging (SSI) system.

The basic real-time task for mission operations involves comparing incoming engineering telemetry to a combination of predicted data values and limit ranges. Specific predictions reflect subsystem goals that result from the planned sequence of subsystem events, and the limit ranges reflect the general operating parameters of the instrument. This task involves two AI components: intelligent input data management and knowledge-based anomaly detection/analysis, in addition to the basic real-time monitoring task. Here we focus on the first of these. The (competing) goals of intelligent data management in this application are to dynamically adjust input data volumes to meet the processing capabilities of the host hardware, while maximizing the information content, maintaining alertness to unusual events in the input data, and focusing on particularly relevant tasks. The particular trade-off we examine in this paper to illustrate our technique is a timeliness trade-off: representativeness of the input data versus timeliness of the solution.

In SSI, four possible data management alternatives have been specified as a result of extensive interviews with an imaging subsystem specialist as part of the first step of DTE. These alternatives are: eliminating channels not in the basic monitoring set, eliminating channels not in the minimal set, reducing sampling rate on heuristically defined subset of channels, and reducing sampling rate on the entire channel set. The converse set of alternatives applies when data rates or computational load from other processes decrease. These converse alternatives include adding channels in the full monitoring set, adding channels in the basic set, increasing sampling rate on a selected channel subset, and increasing sampling rate on the entire channel set.

The four specified alternatives (numbered 1.1, 1.2, 2.1, and 2.2 respectively) are evaluated with regard to three criteria that define data representativeness. For data reduction, these include: (A) non-dynamic behavior, (B) irrelevance to an existing problem area, and (C) non-negative impact on monitoring integrity. A data channel must exhibit non-dynamic behavior before it can be eliminated; frequent channel value changes indicate a high level of activity that must be monitored to maintain adequate representativeness. When representativeness is an issue, irrelevance to existing problem areas is important in deciding which channels to remove from the monitored set. Finally, only channels that do not compromise monitoring process integrity in current circumstances can be eliminated without impacting representativeness. Conversely, when the size of the monitoring set is being increased, the criteria must become (A) dynamic behavior, (B) relevance to an existing problem area, and (C) positive impact on monitoring integrity.

The second step also requires the specification of domain knowledge that shows how to implement the alternatives. In SSI, the channel elimination alternatives and the second sampling rate alternative are influenced most heavily by a decision tree that defines deletable channel subsets and the circumstances under which they apply. There are also exceptions that apply to some deletable subsets with respect to criterion (A). This exception arises because channels with a significant level of activity should not be eliminated from the monitored set even if they are part of an appropriate deletable subset. In contrast, the heuristically-defined sampling rate alternative is entirely governed by the specific situation in which it is applied. In a *normal operating mode*, the sampling rate can be reduced on all channels that are not part of the critical subset. In an *anomaly detection mode*, the sampling rate should only be reduced on channels that are irrelevant to anomaly detection. However, in the event of large backlogs, reduction on sampling of all channels may be desirable.

Occasionally channels must be added irrespective of timeliness. This is because in anomaly detection mode, increased representativeness takes instant precedence, and channels pertinent to that anomaly are added. With multiple simultaneous anomalies, more channels may be needed. Subsequently, timeliness considerations may be applied to some of the other channels in the monitoring set. When the system returns to a normal operating mode, the channels relevant to a previously resolved anomaly may be candidates for removal from the monitoring set if timeliness must be improved.

In the third step, relative weights are assigned to the attributes. Initial weights and variance ranges for these weights are defined so the weights can be adjusted during the reasoning process. This allows the weights to accommodate changing circumstances in the monitored environment. Weight variations are initiated when the system detects that its performance is degrading, and are implemented using rules that provide updates based on situational parameters. This step also entails subjectively ranking each alternative in the context of each criterion at design time, as shown in Figure 1. The ranking, obtained and checked for consistency with the help of the subsystem expert, is on a scale of 0 to 100 (with 100 having the maximum value). For example, alternative 2.1 obviously ranks the highest with regard to B, because the expert specifically designed this alternative not to impact channels with relevance to an existing problem area. Alternative 1.1, which removes the largest number of anomaly-related channels, is perceived to be the poorest choice with regard to criterion B. Conversely, when judged against criterion C, alternative 1.1 has the highest ranking because the channels that it removes generally are the first to be removed and are only added back in small subsets in the event of anomalies. Two

sets of weights are defined for this application, as shown in Figure 2. The first set applies in normal operating mode and the second applies in anomaly detection mode. In normal operating mode, irrelevance of a channel to an existing problem area is given no weight, because no problems are present. However, in anomaly analysis mode, this attribute receives the greatest weight.

	ALTERNATIVE NUMBER			
ATTRIBUTE	1.1	1.2	2.1	2.2
A	75	90	30	40
B	20	30	90	50
C	100	75	40	25

Figure 1. Values of the Alternatives for the Galileo SSI Trade-off.

In the fourth step, the single-attribute alternative rankings and the attribute weights are aggregated into an overall evaluation of alternatives which combines with the application-specific domain knowledge to enable the selection of most valuable alternative for the given circumstances. This step differs most significantly from the comparable static SEU step for two reasons. First, circumstances dictate varying weights, which in turn dictate varying aggregations. Secondly, circumstances may vary the knowledge that is applied from situation to situation. Examples of the varying aggregations that are obtained for both operating modes are shown in the tables of Figure 3. These tables show that the data management actions that are most compatible with maintaining maximum representativeness are determined by external circumstances. The ranking of the four alternatives with regard to representativeness value in varying circumstances is summarized in Figure 4, with 1 being the highest ranking and 4 being the lowest.

A ——— NON-DYNAMIC BEHAVIOR ——— N.O.M. (.45+/- 0.2) / A.D.M. (.15 +/-0.1)

B ——— IRRELEVANCE TO AN EXISTING PROBLEM AREA ——— N.O.M. (0.0) / A.D.M. (.60)

C ——— NON-NEGATIVE IMPACT ON MONITORING INTEGRITY ——— N.O.M. (.15 +/- 0.2) / A.D.M. (.25 +/- 0.1)

(N.O.M. - Normal Operation Mode / A.D.M.- Anomaly Detection Mode)

Figure 2. Attributes and Weights for Telemetry Reduction in the Galileo SSI Trade-off.

The final activity of this procedure involves the selection of an alternative based on dynamic evaluation of the representativeness vs. timeliness trade-off. In order to make this trade-off, the four alternatives must also be evaluated with regard to timeliness. The timeliness impact of an alternative is directly proportional to the percentage reduction (or increase) in the number of monitored channels that results from implementing that alternative. However, this percentage must be calculated immediately prior to making the trade-off, based on the number of channels in the current set, because the number of monitored channels is a dynamic quantity determined by the set of events leading up to the current circumstances. To clarify the dynamic and adaptive nature of this evaluation, we consider the following example.

Assume that the monitoring system has just been brought on-line. Initially, all 49 channels are in the monitored set. After some time the system detects that an input backlog is building, and responds by deciding that some channels must be removed from the monitored set. No anomalies have been detected as yet, and no modifications to the starting weights have been suggested by the knowledge base. As a result of this situation, the system finds itself using the aggregate values in the first line of Figure 3 (top) as representativeness values.

Timeliness values are obtained by calculating the net percentage reduction in input data. Alternative 1.1 eliminates the channels not in the minimal set, or 32 of the 49 channels. Alternative 1.2 eliminates the channels not in the basic set, or 24 of the 49 channels (as governed by domain rules that are not discussed in detail here). These alternatives therefore result in a 65% and a 50% reduction respectively. According to our heuristics, the reduced sampling alternatives can eliminate 4 out of every 5 input values when no anomalies are present. Thus, with alternative 2.1, we can eliminate 80% of a subset of the monitored set. Under the present circumstances, this subset consists of all channels not in the basic set. A reduction of 80% is therefore possible on 24 of the 49 channels. With alternative 2.2, we eliminate 80% of the sampling on the entire channel set, resulting in reductions of 50% and 80% respectively. The percentage reductions are plotted against the aggregate representativeness value for each alternative as shown in Figure 5 (left). Both representativeness and timeliness are thus rated on a scale of 0-100; one unit on the representativeness scale is equivalent to one unit on the timeliness scale. The indifference curves shown in the figure are implied by this constant trade-off of units, alternatives lying on the same indifference curve have equivalent value, and alternatives lying nearest to the upper right of the graph are perceived as best. For this application, the alternatives in order of preference are 1.1, 1.2, 2.2 and 2.1. (Note that timeliness considerations have changed the order of preference from that shown in Figure 4, which is based on representativeness alone.) As a result of this analysis, alternative 1.1 is selected and implemented. Our system is now actively monitoring only 17 of the 49 channels, and is achieving adequate throughput. We will assume that at some later time, an anomaly appears on channel 1910, which requires three additional channels to be added.

				ALTERNATIVE NUMBER			
				1.1	1.2	2.1	2.2
Attribute	*Weight**	*Weight***	*Weight****				
A	*0.45*	*0.65*	*0.25*	75	90	30	40
B	*0*	*0*	*0*	20	30	90	50
C	*0. 55*	*0.35*	*0.75*	100	75	40	25
Aggregate Value (using weight*)*				88.57	81.75	35.5	31.75
*Aggregate Value** (using weight**)*				83.75	84.75	33.5	34.75
*Aggregate Value*** (using weight***)*				93.75	78.75	37.5	28.75

* N.O.M. with no modification on starting weights
** N.O.M. with weight modification for greater emphasis on environmental dynamics
*** N.O.M. with weight modification for greater empahsis on overall monitoring integrity

				ALTERNATIVE NUMBER			
				1.1	1.2	2.1	2.2
Attribute	*Weight**	*Weight***	*Weight****				
A	*0.15*	*0.25*	*0.05*	75	90	30	40
B	*0.6*	*0.6*	*0*	20	30	90	50
C	*0. 25*	*0.15*	*0.35*	100	75	40	25
Aggregate Value (using weight*)*				48.25	49.75	68.5	42.25
*Aggregate Value** (using weight**)*				45.75	51.75	67.5	43.75
*Aggregate Value*** (using wieght***)*				50.75	48.75	69.5	33.25

* A.D.M. with no modification on starting weights
** A.D.M. with weight modification for greater emphasis on environmental dynamics
*** A.D.M. with weight modification for greater empahsis on overall monitoring integrity

Figure 3. Aggregate Values of Alternatives for Varying Weights in Normal Operational Mode (top) and Anomaly Detection Mode (bottom).

ALTERNATIVE / MODE	Elimination of chan. not in basic subset	Elimination of chan. not in critical subset	Sampling reduction on heuristic subset	Sampling reduction on entire subset
N.O.M. with no modification	1	2	3	4
N.O.M. with backlog modification	2	1	4	3
N.O.M. with monitoring modification	1	2	3	4
A.D.M. with no modification	3	2	1	4
A.D.M. with backlog modification	3	2	1	4
A.D.M. withmonitoring modification	2	3	1	4

Figure 4. Rankings of data management alternative values with respect to representativeness for various modes.

The anomaly is solved, and at some later time, another anomaly appears on channel 1881, requiring the addition of 12 more channels.

We are now actively monitoring 32 channels, and are beginning to build a backlog. The system's backlog detection module detects the backlog, and initiates meta-reasoning to reduce it. Figure 5 (right) shows the re-evaluation in response to the environment change at this point. The analysis will have been as follows. Alternatives 1.1 and 1.2 will both allow only 3 channels of the 32 channels in the monitored set to be eliminated. This is because 12 of the channels pertain to the current anomaly and 17 belong to the minimal set. Thus, both alternatives achieve a 9.3% reduction in input data. Alternative 2.1 reduces sampling on approximately 60% of the channels in the sampling set, but because we are in the anomaly detection mode, we now only filter half of the input data from these channels, achieving an effective reduction of 30%. With alternative 2.2, we filter half of the input data on all 32 channels for an effective reduction of 50%. These values are plotted against representativeness as shown in Figure 5. In this case, however, the selection of an alternative s not as obvious as in the previous iteration; alternatives 2.1 and 2.2 are very close to lying on the same

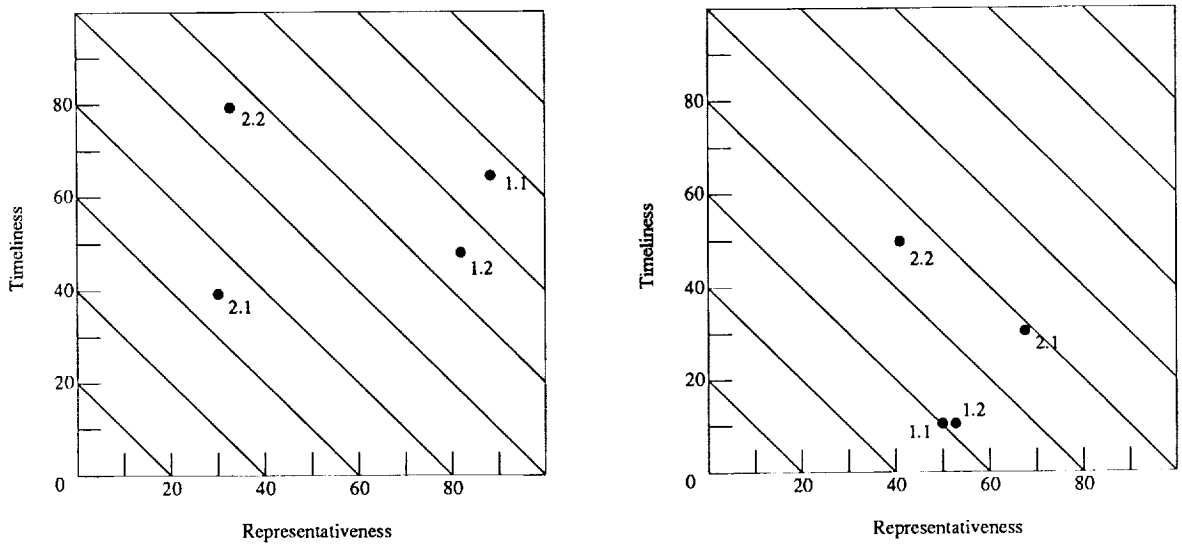


Figure 5. Timeliness vs. Representativeness for Dynamic Input Data Management in Galileo SSI Monitoring. Figure (a), at the right , shows the evaluation of the tradeoff that is made at system initialization, and Figure (b), at the left, shows the how the dynamics of the environment change after several hypothetical anomalies have been detected.

indifference curve. However, heuristics indicate that in the anomaly detection mode, representativeness is the more important consideration, and alternative 2.1 must be selected. Eventually, the anomaly on channel 1881 is resolved, and we return to the normal operation mode. Assuming no change in data rate, in this mode a similar analysis will cause the system to return to its original choice of alternative 1.1, and to continue fully monitoring only channels in the basic subset.

This example has shown the effectiveness of combining decision theory with heuristics to dynamically make real-time trade-offs for intelligent data management. The example illustrates the dynamic nature of the decision environment, and demonstrates the ability to use domain specific heuristics to guide the trade-off process and achieve real-time meta-reasoning for run-time control.

4. Conclusions

Several approaches are known for adapting AI problem solving to dynamic changes in system operating conditions, but simultaneously adapting decision criteria in a principled way has been difficult. This paper has described a general technique for dynamically making performance trade-offs to achieve these ends using a combination of decision theory and heuristic domain knowledge. Dynamic Tradeoff Evaluation is a knowledge-based extension of multi-attribute utility theory. In DTE, multi-attribute utility theory is used to rank alternatives in a preference space and heuristic decision rules are used at run-time to dynamically re-weight the attributes that govern the value of individual alternatives. This enables dynamic selection among preference criteria in the preference space, depending on situational attributes and operational modes. DTE is suitable for highly dynamic real-time environments, as illustrated by its application in specific trade-offs spacecraft telemetry monitoring. It provides a new, rigorous, and effective way to simultaneously adapt system decision criteria and problem-solving parameters.

5. Acknowledgment

The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The domain knowledge pertaining to mission operations and solid state imaging that was used in this research was contributed by William Cunningham at the Jet Propulsion Laboratory.

6. References

D'Ambrosio 1990] B. D'Ambrosio. Constrained Rational Agency. To appear in IEEE Transactions on Systems, Man, and Cybernetics. 1990.

[Dyer 1979] J. S. Dyer and R. A. Sarin. Measurable Multi-attribute Value Functions. Operations Research, 22, 810-822.

[Edwards 1977] W. Edwards. How to Use Multi-attribute Utility Measurement for Social Decision Making. IEEE Transactions on Systems, Man and Cybernetics, SMC-7, 326-40.

[Horvitz 1988] E. J. Horvitz, J. S. Breese, and M. Henrion. Decision Theory in Expert Systems and Artificial Intelligence. International Journal Of Approximate Reasoning, Special Issue on Uncertain Reasoning: 247-302, 1988.

[Horvitz 1989] E. J. Horvitz, G. F. Cooper, and D. E. Heckerman. Reflection and Action Under Scarce Resources: Theoretical Principles and Empirical Study. Proceedings of the International Joint Conference on Artificial Intelligence. Detroit, MI 1989.

[Keeney 1976] R. L. Keeney and A. Sicherman. An Interactive Computer Program for Assessing and Using Multi-attribute Utility Functions. Behavioral Science, 21, 173-182.

[Laffey 1988] T. J. Laffey, S. Weitzenkamp, J. Read, S. Kao, and J. Schmidt. Intelligent Real-time Monitoring. Proceedings of National Conference on Artificial Intelligence, 1988.

[Muratore 1990] J. F. Muratore, T. A. Heindel, T. B. Murphy, A. N. Rasmussen, R. Z. McFarland. Acquisition at Mission Control. Communications of the ACM, 33:12. December 1990.

[Pearl 1988] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, San Mateo, CA., 1988.

[Russell 1989] S. J. Russell and E. H. Wefald. On Optimal Game-Tree Search Using Rational Meta-reasoning, International Joint Conference on Artificial Intelligence, Detroit, Michigan, 1989, 334-340.

[Shachter 1987] R. D. Shachter and D. E. Heckerman. Thinking Backward for Knowledge Acquisition. AI Magazine, 8, 55-63, 1987.

[Schwuttke 1990] U. M. Schwuttke, J. R. Veregge, R. Angelino, C. L. Childs. MARVEL: An Automated Productivity Enhancement Tool for Multi-mission and Multi-subsystem Spacecraft Operations. First International Symposium on Ground Data Systems for Spacecraft Control. Darmstadt, W. Germany. 1990.

[Schwuttke 1991] U. M. Schwuttke. Intelligent Real-time Monitoring of Complex Systems. Ph.D. Thesis in progress. University of Southern California., Los Angeles, CA. 1991.

[Simon 1955] H. A. Simon. A Behavioral Model of Rational Choice. Quart. J. Econ., 69, 99-118.

[von Winterfeldt 1986] D. von Winterfeldt and W. Edwards. Decision Analysis and Behavioral Research. Cambridge University Press. 1986.

Session I4: ICAT

Session Chair: Jim Fleming

N93-11937

On the Acquisition and Representation of Procedural Knowledge

T. Saito*, C. Ortiz, and R.B. Loftin**
Software Technology Branch (PT4)
NASA/Johnson Space Center
Houston, Texas

Historically knowledge acquisition has proven to be one of the greatest barriers to the development of intelligent systems. Current practice generally requires lengthy interactions between the expert whose knowledge is to be captured and the knowledge engineer whose responsibility is to acquire and represent the expert's knowledge in a useful form. Although much research has been devoted to the development of methodologies and computer software to aid in the capture and representation of some types of knowledge, little attention has been devoted to procedural knowledge. NASA personnel, on the other hand, frequently perform tasks that are primarily procedural in nature. In this paper we will review previous work in the field of knowledge acquisition and then focus on knowledge acquisition for procedural tasks with special attention devoted to the Navy's VISTA*** tool. We will describe the design and development of a system for the acquisition and representation of procedural knowledge—TARGET (Task Analysis and Rule Generation Tool). TARGET is intended as a tool that permits experts to visually describe procedural tasks and as a common medium for knowledge refinement by the expert and knowledge engineer. The system is designed to represent the acquired knowledge in the form of production rules. Systems such as TARGET have the potential to profoundly reduce the time, difficulties, and costs of developing knowledge-based systems for the performance of procedural tasks.

Introduction

In order to set the stage for a description of "yet another" knowledge acquisition tool a brief review of our evaluation of many existing knowledge acquisition tools is presented below . Following this review a Navy-developed program for task analysis is discussed in some detail. The remainder of the paper is devoted to a description of our own work in creating a knowledge acquisition tool specifically for procedural tasks. In addition to its role in aiding experts and knowledge engineers in the process of knowledge acquisition, this tool also provides mechanisms to support knowledge refinement and its representation in the form of production rules.

Processes and software designed to aid knowledge acquisition can be characterized by the nature of their delivery and implementation methods and styles as well as their ability to extract knowledge. As with other types of software tools and products, knowledge acquisition tools, either on the market or under development, seem to come in many "flavors", "colors," and "shapes". Knowledge acquisition tools operate as front-ends as well as embedded modules within existing expert systems and/or expert system development packages. Delivery platforms range from PCs and Macintosh® computers to RISC and symbolic workstations to IBM® plug-compatible mainframe systems. As the size of the platform grows, so often does the complexity and sophistication of the knowledge acquisition tool or module that it supports. Such commercial PC-based tools as GURU® (mdbs, Inc.), VP-Expert™ (Paperback Software), and Socrates™ (CIM Solutions) possess modest levels of knowledge acquisition features generally oriented toward knowledge representation in the form of rules. On the other hand, mainframe compatible ADS® (Aion Development System from Aion Corp.), Mercury KBE™ (Knowledge Base Enterprise from Artificial Intelligence Technologies) and Knowledge Shaper (Perceptrics, Inc.) all provide more elaborate knowledge acquisition functions and features (e.g., object-oriented representation and management) than the smaller systems. Tools under development such as KART (Knowledge Acquisition Reasoning Tool from IBM) and AKAT (Automated Knowledge Acquisition Tool from Harris Corp.) represent tools that operate within the less exotic workstation environments. However, other institutional knowledge acquisition tools such as Aquinas (Boeing) and KNACK (Carnegie-Mellon) provide sophisticated examples of the future of knowledge acquisition tools. DART (Design Alternative Reasoning Tool - Boeing ATC) is one spinoff tool whose design architecture was derived from its more renowned predecessor, Aquinas.

Operating system environments range from Microsoft DOS® (Disk Operating System) to the Macintosh® OS to UNIX® to IBM's VM® (Virtual Memory) and MVS® (Multi-Virtual System). Some knowledge acquisition tool developers have released their tools under several operating systems. Nextra, a commercial tool marketed by Neuron Data as a front end to the Nexpert Object expert system, functions under UNIX®, VM® (Virtual Machine - IBM), VMS® (Virtual Machine System from DEC®) and Macintosh® operating systems. Other tools, as they evolve, may well follow these same implementation and delivery strategies.

Various authoring tools have evolved to solve the problems associated with the creation of a specific expert system.[1]. Originally, most knowledge-acquisition-oriented tool designs were directed toward rating or categorizing problems or knowledge. To capture specific knowledge, the developer distinguishes between types of knowledge methods/approaches. Although sharing many of the same goals, the existing methodologies are numerous—ranging from frame modeling to case-based reasoning models to repertory-grid rating structures. The various knowledge types, addressed by these systems, range from semantic/taxonomic to declarative to procedural, affecting the design and performance decisions of researchers and implementers[2]. Knowledge representations, including frames, objects, rules and decision trees, are used to capture and execute

expertise. At this point, most would agree that no one tool accommodates all of the cognitive styles needed to gather the information/knowledge necessary for the creation of an expert system in one contiguous process. It is clear that viable standards have yet to be fully established and accepted.

To further complicate the issue, getting a subject matter expert's (SME) attention, time/commitment, help, and data sources is usually difficult at best. SMEs tend to differ in their communication abilities and styles, willingness to cooperate/availability, and degree of computer literacy, potentially affecting the overall success of the knowledge acquisition process[3]. The strategy of providing the SME with a tool that can be used to document his mission(s) or task(s), on his own and within his schedule, would resolve some of the traditional headaches associated with a knowledge engineer constantly "hovering over" an SME. However, the disadvantages of such a strategy may be the lack of positive reinforcement or external motivation (i.e., SMEs might put off documenting their task/mission unless periodically reminded or encouraged.).

As computer hardware power evolves, more latitude in presentation methods will be available. Visual conception and communication of abstract information will become more common. The strategic fusion of graphical display (bit-map, meta-graphic, etc.) and graphical input device (mouse, light-pen, trackball, etc.) technologies will facilitate visual, as well as textual representation, of knowledge[4]. Drawing tools already allow the user to produce and manipulate complex graphics. The role of these tools can also combine with organizational algorithms to create more intelligent diagrams, flow charts, interactive decision trees, etc. With users becoming more adept at using systems with pictorial modeling capabilities, the mode of knowledge acquisition will also benefit from such advances.

Procedural knowledge acquisition via task analysis is a reasonable candidate for graphical representation modes. Decomposing a complex set of steps that makes up a specific mission or task requires cognitive visualization and the ability to formulate and reformulate the decomposition of those steps or actions. The specific heuristic procedures that most SMEs employ share certain levels of organization and recall[5]. The path in which a procedure evolves starts with specific agendas and goals. The last or final action of reaching or satisfying those actual goals would end the procedure. On the other hand, any actions that would restart a process (loop) would occur before the goal oriented or last action. Decisions may be made during a task that direct the expert along alternative paths which may or may not be taken in other performances of the same task. In cases where the processes offer one or more options to complete a task, the process diverges into as many paths necessary to meet the optional requirements. Each path would contain specific values for technique evaluation or other modes of feedback. These types of complexities lend themselves to representation in a visual form. Below we explore two attempts to provide just such a visual metaphor for knowledge acquisition and representation.

VISTA: The Graphical Predecessor

Of over twenty expert systems assessed by NASA/Johnson Space Center, a graphically-oriented task analysis tool developed by Robert Ahlers of the Naval Training Systems Center (NTSC) showed the most promise for addressing procedural knowledge acquisition. VISTA (Visual Interactive System for Task Analysis) has proven to be a tool an expert can use to easily define and document specified tasks in a "comfortable" and modifiable form.

The NTSC in Orlando, Fl., participating in a governmental tri-service project with the Air Force Human Resources Lab (HRL) and the Army Research Institute (ARI), directed a project that produced a prototype knowledge acquisition tool called KA-1 (Knowledge Acquisition-1). KA-1 was first designed and implemented in Lisp within the Symbolics/Genera 6.x environment. Recognizing the appropriateness of implementing their own knowledge acquisition strategies to acquire task or procedural knowledge, NTSC, after the end of the KA-1 project, started work on its first prototype (named AFEAT—Automated Front-End Analysis Tool). NTSC later redesigned and released an enhanced version, renaming the knowledge acquisition tool VISTA. Both of these applications have been developed on PC platforms using Smalltalk-V.

The NTSC designed VISTA to compose task lists and hierarchies from a graphical representation of a knowledge base. VISTA identifies subsets of tasks meeting specified selection criteria, training objectives, and/or personal performance profiles. VISTA's strategy, permitting the SME to establish task hierarchies and relationships, yields a final report of procedural step data with corresponding conditions and criteria.

The VISTA system is a graphical user interface (GUI) oriented tool that builds box-flow style representations that a user can utilize to document various task levels. VISTA is essentially a qualitative analysis tool directed toward task and procedure decomposition into their component parts, in a largely top-down style. In knowledge acquisition mode, the knowledge engineer and SME could conduct the decomposition process together or the SME could essentially use the tool without direct knowledge engineer support. The system also has a knack for allowing a group of experts to huddle in front of a VISTA screen for consensus verification and modification.

VISTA maintains a fragile balance between ease of use and design complexity/intricacy. Although VISTA does not possess the "bells and whistles" of more sophisticated systems like Aquinas and Protege, it provides enough knowledge modeling (procedural/declarative) support to allow the SME or knowledge engineer to build a fairly elaborate knowledge base without sacrificing the attractiveness of its user interface.

VISTA provides a "windows-icon-mouse-pointing" (christened, WIMP) interface environment based on a grid-marked Work Area in which the user builds task networks. The WIMP approach facilitates the rapid selection and execution of system functions to minimize user keystrokes. The Work Area is lined on three sides with icon and menu selectable functions:

1) System Command Menu Bar (top)
2) Function/Graphics Icons (left side)
3) Message/Explanation Bar (bottom)

Although VISTA provides no bona fide compilation facilities to check the knowledge base for completeness or accuracy, it does utilize some of the Smalltalk inspection and reporting features to help the user confirm the knowledge input into the system. VISTA provides a windowed environment through which decomposition can be organized and recorded. Ultimately the user, knowledge engineer or SME, is responsible for the overall quality checking of the knowledge base before its representation in or transfer to other applications. VISTA's report facilities offer some assistance in this quality checking process. Reports can be generated to provide moderately high-level feedback to the knowledge engineer and SME. VISTA produces the following reports:

- Hierarchical Statistics: counts nodes and subnets at each level
- Task hierarchy: keeps a sequential/hierarchical account of tasks
- Input grammar: maintains various types of component titles in categorized form
- Notecards: keeps notes on conditions, states or other user-supplied details
- Highlighted tasks: accounts for subnet levels

VISTA supports the identification and conceptualization phases of knowledge acquisition with its network approach to knowledge representation. Duties, tasks/subtasks, or steps/substeps within a process can be defined, documented and structured to reflect these relationships to other duties, tasks/subtasks, or steps/substeps.

Given its developmental state, VISTA provides a fairly comprehensive mechanism for generating simple representations at the very first knowledge acquisition session. The next sessions may be used to embellish what has already been elicited, or to create new or modified versions of the knowledge base. The VISTA knowledge acquisition interface gives the user the freedom to generate as complex a hierarchy of knowledge as necessary. However, the disadvantage to such freedom is the ability to create a completely abstract knowledge base with relatively little standards for input. Some guiding controls from the VISTA interface could provide structure to the knowledge acquisition process and greatly enhance the ability of the user to create a "useful" knowledge base.

Although VISTA has a grammar component within its facilities, its ability to correlate domain terms and/or concepts is limited. The open-endedness of the tool design allows key domain definitions to be specifically addressed within a notecard-like management facility and/or defined in a box as a subpart within a process. VISTA does not offer a true lexical facility or natural language interface to accommodate concept definitions.

In addition to the creation of new knowledge base versions, the system also offers the ability to mesh existing VISTA knowledge bases into the current version for incremental enhancements. As new ideas are evolved, the VISTA interface allows the integration of old and new knowledge bases for processing. Task progressions can be devised from scratch or from existing task lists or progressions. A VISTA grammar (task verbs and verb-objects) must be defined in order for new tasks to be created. Creating a grammar from scratch involves building and naming a number of tasks. A significant improvement would support the automatic extraction of a grammar from an existing task description and its integration into a new or different network.

The TARGET Approach

NASA Environment and Needs

The National Aeronautics and Space Administration commits large funding and manpower effort to training new and existing personnel. New recruits are trained to carry out tasks for which they were hired. Existing staff must be trained or retrained to upgrade/update abilities to perform current and new tasks. Significant numbers of training methodologies are utilized involving training manuals, formal classes, instructional computer programs, simulations and on-the-job training. On-the-job training is usually the most effective training mechanism for the more complex tasks requiring substantial autonomy on the part of the task performer. However, this training style is also the most expensive and impractical where trainees significantly outnumber experienced staff.

The effort of educating and training NASA astronauts, flight controllers, and other ground support personnel has generally required extensive on-the-job experience in order for individuals to acquire the knowledge and skills necessary for acceptable performance and/or certification. Current flight schedules, combined with the loss of experienced personnel to retirement/transfer, have significantly reduced the ability of traditional training techniques to produce an adequate number of trained personnel[6]. Recently, it has been shown that workstation-based, intelligent computer-aided training (ICAT) systems can deliver intensive, personalized training to large numbers of trainees, independent of integrated simulations[7]. Such systems can noticeably reduce the amount of training time needed to achieve acceptable levels of performance. After over four years of experience in building such systems, the developers have concluded that the greatest barrier to the large-scale, efficient production of ICAT systems is the extent and difficulty of the knowledge acquisition process. For some time an effort has been underway to create a software tool to aid in the capture of mission support procedural knowledge both to preserve the existing corporate

knowledge base and to assist in the development of decision support expert systems and ICAT systems.

The remainder of this paper details the design and implementation of a knowledge acquisition system tailored to the acquisition and representation of procedural knowledge associated with the performance of complex tasks. The goal of this effort has been the production of a system with an easy-to-learn and "comfortable" user interface that provides powerful mechanisms for the visual expression of procedural knowledge. The ultimate goal of this work is the expression of acquired knowledge in the form of production rules to facilitate the use of the acquired knowledge in expert systems for mission support and training.

TARGET and Design Strategy

Attempting to strike that delicate balance between nonprogrammer usability, design sophistication, and hardware universality, the Task Analysis Rule GEnerating Tool (TARGET) is designed to provide a knowledge acquisition environment for users of commonly-available computer systems (IBM® PCs and Apple© Macintoshes®). The forte of TARGET is the gathering of task or procedural knowledge to be expressed and analyzed graphically as well as contextually. TARGET provides users the ability to graphically decompose a task or mission using a box-flow presentation/manipulation style within a windowed environment.

TARGET is designed to let the SME to start documenting their job or task with minimal training in its use and no absolute need for knowledge engineer intervention. If the SME is not able to find time to work on the knowledge acquisition process alone, TARGET does allow the knowledge engineer and SME to work together in iterative sessions. TARGET is tailored to accommodate a wide range of users, from the novice to the expert. With TARGET, users can develop a discrete representation of tasks and their subtasks within their domains[8]. The system then manages the information entered and represents the knowledge in a "top-down" reporting format that can then be used for rule induction and generation.

Action Descriptions within TARGET

Within the NASA/Johnson Space Center environment, CLIPS (C-Language Integrated Production System) is widely used as an expert system development and delivery vehicle. In order to support the development of intelligent computer-aided training (ICAT) systems, TARGET will implement its rule representations using the rule types and structures originally developed for ICAT systems.

Within the ICAT metaphor the overall mission or task is decomposed into sets of tasks/subtasks that are termed actions. For most effective use, actions are expressed, within reason, at the lowest possible level. At any point in an ICAT training session, the expert expects the trainee to perform a valid action. Each valid action, as defined by the expert, is represented as a CLIPS fact in the following pattern:

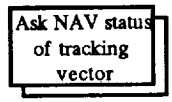
(message-E-to-I <step number> <action type> <argument> <argument> ...)

An action itself comprises at least two <argument> fields that define one single action decomposed into a hierarchical structure of two or more subactions of the form (<action> <argument>). Each <argument> may itself be an <action> at the next lower level. For example, (arg1 arg2 ... argn) can be decomposed into at least two levels: (arg1 arg2) and (arg2 . . . argn). The first pair, (arg1 arg2), is an (<action> <argument>) pair at the top level where the action arg1 has one argument, arg2. In turn, (arg2 ... argn) is another (<action> <argument>) pair at the second level where arg2 has one or more arguments, depending on the value of n. The structure for each action may be different and the number of arguments that belong with each action is variable. The expert is free to decompose the actions and arguments into hierarchies that fit his or her specific domain.

TARGET supports three action types: required, optional and flexible (as defined in the ICAT architecture). Required actions are necessary task(s) and/or subtask(s) performed at specific points in a mission or task. These actions are then asserted to the factlist by the expert module. The following pattern reflects the expected action at a given point in time:

(message-E-to-I <step number> require <argument> <argument> ...)

For example, Figure 1 shows the TARGET visual representation of the action: "Ask the Navigation Officer to give the status (good or bad) of the current state vector (describing a specific orbiting vehicle) obtained from ground- or space-based tracking stations." Following this representation is the CLIPS fact produced to represent the same knowledge.



(message-E-to-I 150 require request nav nav-status)
Fig. 1 Required Task

The ICAT architecture prohibits the performance of further actions if the current action cannot be achieved. In cases where more than one alternative precedes a required action, the alternatives may not all be equivalent. Although they accomplish the same goal, one method may be more suitable than another in a given context. TARGET distinguishes between these alternatives by labeling the "less desirable" as an "other" required action, as opposed to the previous "require" action. In the ICAT architecture, only one "require" action, at any point, exists, with the rest, if any, being "other" actions. When more than one alternative exists for a required action (and none are more suitable than the others), all alternatives are labeled as "other" actions. Other actions are represented by TARGET as:

(message-E-to-I <step number> other <argument> <argument> ...)

Optional actions are those recommended by the SME as good action or technique preferences in a given context. However, the trainee is not prohibited from advancing in the training session if the action is disregarded. Optional actions should not be confused with situations where more than one required action exists at a given time. Optional actions are represented by the following pattern:

(message-E-to-I <step number> optional <argument> <argument> ...)

Figure 2 shows the TARGET visual representation of an optional task.

Fig. 2 Optional Task

Since optional actions do not halt the training session within the ICAT architecture, even if they are ignored by the trainee, <step number> could be any future step beginning from the current context. For example, if the current step is 20 and the expert asserts (message-E-to-I 50 optional ...), that particular optional action remains valid from step 20 until step 50. Validity checks for certain values can be displayed in the user interface. For example, a fact of the form

(message-E-to-I 260 optional check-display vector-comp MET)

indicates that the expert recommends that the Mission Elapsed Time field within the Vector Comparison Table display be checked at any time from the current step through step number 260.

TARGET will internally manage flexible actions, i.e., actions that must be completed by predefined times (similar to a required action but operable over a span of time or steps). Flexible actions operate identically to optional actions inside specified range of steps. If the deadline arrives and the flexible action still has not been completed, it will be changed into a required action. Flexible actions take the following pattern:

(message-E-to-I <step number> flexible <argument> <argument> ...)

TARGET, following the current ICAT architecture, is designed to deal with a series of tasks/subtasks that are performed procedurally or in steps. Steps are defined as the progression from one required action to the next. Steps are represented by numerals and their values increase with the progression of the task.

Task-Action Concepts (Building Blocks)

TARGET employs a free-form flow charting concept. SME or knowledge engineers can explain procedural processes by the use of various icons arranged in the TARGET work area. The tasks can then be linked together using directed arcs to show procedural flow. The task icons are separated into five categories.

The first category consists of actions that are required to complete a process. The required actions are denoted by using a blue (or white, for monochrome monitors) rectangular box on the screen. On the other hand, optional tasks are represented using grey rectangles. The use of color for representing various forms of tasks has been reduced to just two, grey for optional tasks and blue (white) for all others. The shape of the task box is the most important key to determining its function. By using shapes and not colors, TARGET, although designed for color systems, also works reasonably well on monochrome systems.

Fig. 3 Required and Optional Tasks

Hexagonal-shaped structures are used to show processes where a decision is needed. The connections leaving the decision structures are labeled to show what action is to be taken. For example, a decision structure may ask if a process has been completed. This decision may be linked to two other tasks (see Figure 4). The first arc leaving the decision is labeled "YES" and the other is labeled "NO". The arc labels represent the only possible answers allowed by the decision structure.

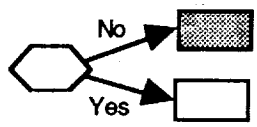


Fig. 4 Decision Task

The fourth basic structure that TARGET provides is a control structure. Control structures are used as a "goto" or looping mechanism. Controls can only jump to other tasks that are on the current layer (see discussion of layers below) and may not jump directly to other controls. Control tasks are denoted by using an ellipse to distinguish them from the other forms of tasks.

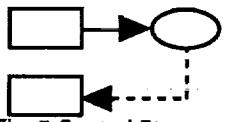


Fig. 5 Control Structure

The fifth task type is called a Discretionary or D-Path (Figure 6). D-Paths, defined as alternate paths when taken, will not affect the final outcome of the procedure. D-Paths are used when decision structures are too strong, but a two or more options exist that can be explored. Eating breakfast is an example of a D-Path. One could eat a large breakfast, a small breakfast or skip breakfast entirely and would not impact the process of "getting ready to go to work".

D-Paths are defined to start when two or more tasks branch off from a single required or optional task. (Decision structures are the only other task structures

that allow for multiple branching.) Tasks within a given D-Path chain must be optional or required tasks and cannot be connected to more than one task at a time. This rule insures that no branching occurs from outside or within the D-path itself. All D-Paths will end at a single common (optional or required) task. The final specification for D-Path structures regards the preference value for taking one choice over another.

A numerical value is given to each D-path and is termed that path's preference value. The preference value is a number that ranks each path choice from highest to lowest preference. The path with the highest numerical value is considered to be the "best" path by the SME. Other paths represent processes that may be performed but are not regarded as optimal by the SME.

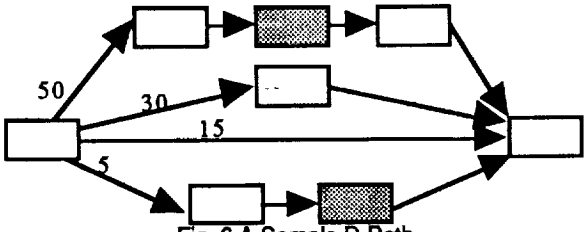


Fig. 6 A Sample D-Path

In addition to the two-dimensional flow, TARGET provides users with the ability to decompose required and optional tasks into a task hierarchy. Task decomposition is used to make complex tasks easier to understand and complicated layers smaller. Tasks that are decomposed are characterized by displaying a shadow and have "child" tasks associated with them. The child tasks are not seen at the "parent" layer but can be found when traversing down the task hierarchy into lower levels (Figure 7).

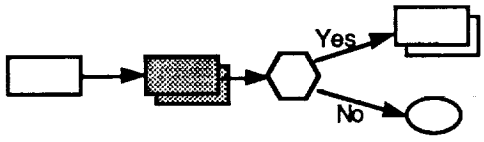


Fig. 7 Task Chain Showing Child Tasks

Navigating through a task hierarchy is done simply by placing the cursor on a leveled task and double-clicking. The screen will clear and be rebuilt showing the selected child tasks. Moving up in the task hierarchy is just as simple, it is done by double-clicking on an area where there is no task. The screen will clear and tasks from the level above will be displayed.

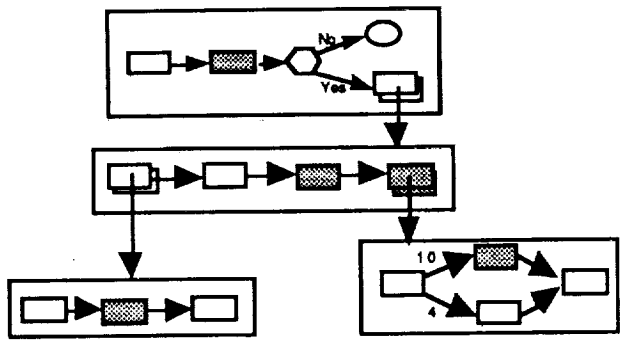


Fig. 8 Decomposition of a Task Hierarchy

Input Features

To facilitate its implementation on many hardware platforms, TARGET was developed around the one-button mouse concept. Using a single mouse button was determined to be the best way of porting the TARGET program to other platforms such as a Macintosh™ computer. Double-clicking the mouse button will move up or down a task hierarchy and single-clicking will activate all other functions. Since single-clicking on a one-button mouse is limiting, TARGET, like VISTA, was designed to be a cursor-driven program.

Users may select a desired function from the main menu, toolbox or keyboard (Figure 9). The cursor will change to reflect the operation that is to be performed. Located on the left-hand side of the screen, the toolbox is one of the most important user interface functions provided by TARGET. Using the toolbox, one can directly manipulate task structures on the current screen by editing, creating, deleting, moving, linking, and un-linking tasks. Help and a display of the task hierarchy may also be accessed through this toolbox.

Menu selections provide an interface for file I/O, for task manipulation, to set user preferences for how TARGET displays the task flow, to control output, and to obtain help. File I/O deals with the reading and writing of TARGET files (.TGT) to and from disk storage. In addition to the .TGT files, .DMN files (generated from VISTA) can also be directly imported. The EDIT menu helps to find tasks as well as permit the moving of tasks up and down the task hierarchy.

The VIEW menu is especially helpful. Using view, one can display tasks in a horizontal or vertical orientation. Optional tasks and the background grid can be hidden or displayed. The entire layout may be redrawn using automatic task placement. HELP provides information regarding TARGET questions as well as an explanation of the help facility itself. OUTPUT provides a mechanism to generate reports or simple rule structures on the screen, printer, or in ASCII format on disks.

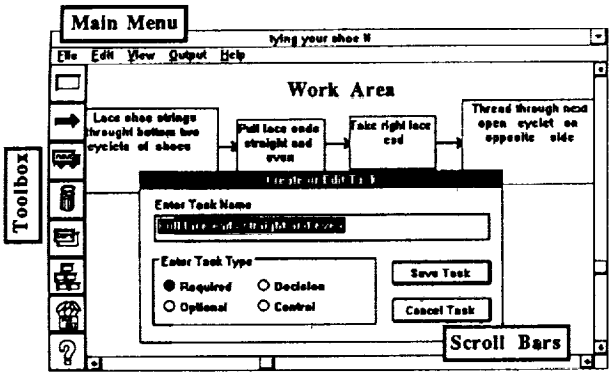


Fig. 9 TARGET User Interface

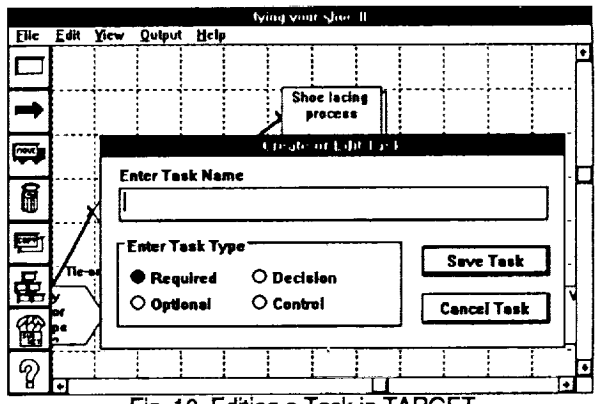


Fig. 10 Editing a Task in TARGET

Documentation Strategies

TARGET's overall visual representation style employs directed graph strategies combined with the box-flow and entity-relationship diagram structures. Task hierarchies require large amounts of layout space while visual analysis is performed[9]. Previously, where most task analysis efforts were done by paper and pencil, the graphical medium provides a facility to organize and manipulate various levels of tasks[10].

Decomposing the task into single steps is the recommended strategy within the TARGET environment. TARGET provides features for organizing selected missions or tasks into distinct hierarchical levels. Individual steps can be laid out and studied interactively by the SME and/or knowledge expert. Immediate graphical feedback for analysis is possible. TARGET will also accommodate the text editing process as well as maintain box/task step order.

The basic human factors rule-of-thumb recommends the "7 plus-or-minus 2" boxes/tasks per screen for creating hierarchical box diagrams. When documenting complex tasks, however, the user should not be limited (especially in his initial effort) to a specific number of boxes[11]. What counts is the ability to keep track of the task networks being developed. Within TARGET, however, there is currently a limit, within a specific level, of 100 boxes.

TARGET's documentation input requirements pose no constraints at this time. The "Enter Task Name" function allows free-form input. However, within the TARGET environment, addressing another person would be the most effective way to phrase a task. Communication with that someone should be done in first person with an implied "you". For example, "You should":

- Turn on light
- Flip switch up
- Report to Commander
- Allocate extended memory to DOS
- Shut down all systems
- Fire missile

Once accustomed to the approach TARGET takes in documenting task flows, the domain expert can establish multiple mission/task levels hierarchically using his or her own classification or categorization strategies to express specific procedures. TARGET provides a traversing mechanism in which the user can create, maintain, and check task subnetworks with minimal keystrokes.

TARGET will provide a template, breaking down the task steps into sequential dependencies, to facilitate the construction of the left-hand and right-hand sides of a CLIPS rule. For example, a step will have to be performed before another step is executed. The previous step becomes the dependency or left-hand condition for the right-hand or current step. A task box will encapsulate all of the necessary information that makes that particular task significant, whether it be required or optional (Figure 12). The user need not be concerned with "keeping score" on these issues, TARGET, through its graphical maintenance facility, will manage the top-down representation and coordination of tasks.

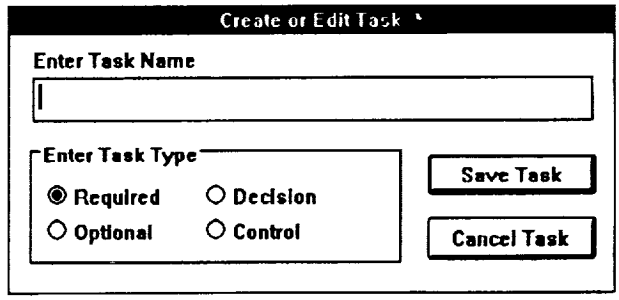


Fig. 11 Create/Edit Dialog

Within the CLIPS rule environment, steps documented in the TARGET style will automatically support the rule composition process. Each step will operate as part of the left-hand or right-hand side of a rule where applicable. TARGET manages context and sequence from the user interface.

As knowledge acquisition iterations progress, the task steps will have to evolve into executable forms of knowledge for implementation within an expert system environment. Generating rule representations induced

from a top-down, sequential list assumes that high-level verification has been performed. TARGET will allow specific levels of manual, visual, and list verifications throughout the task hierarchies. However, with TARGET, verification and validation will also have to be carried out in CLIPS on the generated rule-base since TARGET has no automated verification and validation component at this point.

Rule Control Structure Design

For rule propagation, TARGET will implement the ICAT control structure guidelines where rules employ two parts, the data and the task process. In general, facts will be data equivalents. The process or task step will be manifested in the form of a rule. Facts will be used to specify the current context or environment and to represent the "actions" taken by a user or the expert system. All fact combinations must be matched against rule patterns to determine which rules may fire. Then, one rule would be used to generate potentially executable actions and update the fact base. This process will be repeated until no rules can fire. More specifically, the ICAT control structure revolves around two significant points.

1) Message passing protocols are used by independent rule sets for communication and
2) Tasks are procedural/step-by-step in structure.

Report Generation with TARGET

TARGET provides several reporting mechanisms. The first report that the average user will experience is the graphical representation of tasks in a task hierarchy. Figure 12 represents a sample graphical layer that would be produced by a user in the graphical interface. Task hierarchies in a textual form can also be generated from TARGET to provide another look at the procedural process being described (Figure 13). Files that store all the information TARGET needs to build task networks are called .TGT files (Figure 14). .TGT files are closely related to the .DMN files created by VISTA. The file similarity is intended to expedite the conversion from VISTA files into TARGET representations. The last figure (Figure 15) in this section shows CLIPS rules written in final form produced by TARGET.

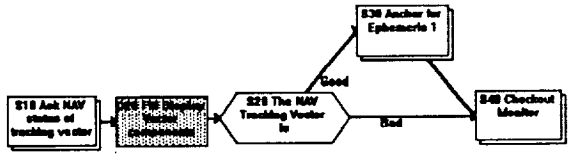


Fig. 12 Sample Task Chain Described Graphically in TARGET

1.0 Request NAV tracking vector status
- 1.1 (f) "toggle-offset-sign"
- 1.2 (f) "enter orbiter PKM offset in Worksheet"
- 1.3 (f) "enter deploy separation rate in Worksheet"
- 1.4 (f) "enter deploy spin axis declination in Worksheet"
- 1.5 (f) "enter PKM offset in Deploy Worksheet"
- 1.6 (f) "enter deploy relative right Ascension in Worksheet"
- 1.7 (f) "enter Z component of deploy delta V in Worksheet"
- 1.8 (f) "inform SDP DYN of payload data"
- 1.9 (f) "enter target PKM offset in Worksheet"

2.0 (o) check Vector Comparison Table
3.0 The NAV Tracking Vector is (Good)
- 4.0 Anchor ephemeris for shuttle
  - 4.1 (o) inspect new ephemerides
  - 4.2 (o) request Vector Comparison Table
  - 4.3 (o) request Trajectory Digitals display for ascending node
  - 4.4 (o) request Trajectory Digitals for descending node

3.0 The NAV Tracking Vector is (Bad)
5.0 request Checkout Monitor w/o Traj Digitals

Fig. 13 Task Hierarchy Description Generated from TARGET

```
REQUIRED
S10 Request NAV tracking vector status
2                                   <Task Key>
273,139                             <X,Y Pos>
1                                   <Parent>
11,10,9,8,7,6,5,4,3                 <Children>
13                                  <Connected_To>
OPTIONAL
O20 Fill Display Vector components
13                                  <Task Key>
402,140                             <X,Y Pos>
1                                   <Parent>
14                                  <Connected_To>
DECISION
S20 The NAV Vector is Good
14                                  <Task Key>
545,136                             <X,Y Pos>
1                                   <Parent>
Good
18                                  <Connected_To>
Bad
12                                  <Connected_To>
REQUIRED
S30 anchor ephemeris for shuttle
```

```
18                          <Task Key>
691,55                      <X,Y Pos>
1                           <Parent>
17,15                       <Children>
12                          <Connected_To>
```

Fig. 14 .TGT File Derived from a VISTA .DMN File

```
(defrule s10-get-nav-info "Request NAV tracking vector
status"
        (step ?s&10)
        (checkpoint expert)
=>
        (assert (message-E-to-I ?s require req-nav get-
        nav-status)
                (next-step 20))
)

(defrule o20-display-vector-comp "Fill Display Vector
components"
        (step ?s&20)
        (checkpoint expert)
 =>
        (assert
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_CUR_TIME)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_CUR_DESC)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_CUR_A)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_3RD_TIME)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_3RD_DESC)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_3RD_A)
        (message-E-to-I ?s optional check-display dis-
        vector-comp VC_3RD_V))
)

(defrule s20-good-nav "The NAV Vector is Good"
        (step ?s&20)
        (environment 0 nav-tracking good)
        (checkpoint expert)
=>
        (assert (message-E-to-I ?s require req-nav put-
        nav-in-slot v39))
        (assert (next-step 30))
)

(defrule s20-bad-nav "The NAV Vector is Bad"
        ?f1 <- (step ?s&20)
        (environment 0 nav-tracking no-good)
        ?f2 <- (last-step ?)
        (checkpoint expert)
=>
        (retract ?f1 ?f2)
        (assert (step 40)
                (last-step ?s))
)
```

Figure 15. TARGET Rule Generated File

Conclusion

TARGET will have the capability to significantly impact the development of ICAT systems as well as the development of other intelligent systems. For any procedural knowledge acquisition task TARGET can enhance the ability of the expert to visualize and organize a task or process. We believe that procedural visualization of this type will become more popular as more tools with organizational diagnosis capabilities evolve[12].

As knowledge acquisition, as a discipline within artificial intelligence, evolves, more tools to assist in the knowledge acquisition process will also become available in useful forms. TARGET, and tools like TARGET, will be employed within their own "niche" and will also be integrated with other methodologies in the future. Although TARGET models currently sequence within the task hierarchy structure for rule induction, we will dedicate additional efforts to encapsulating more peripheral knowledge into the various steps within the network. Particularly, for TARGET, such issues as gathering artifact data, selected action rationale, and interactive verification and validation of rules will be addressed in the future.

---

* Computer Sciences Corporation

..** University of Houston - Downtown Campus
(bloftin@nasamail.nasa.gov)

*** VISTA (Visual Interactive System for Task Analysis)
Naval Systems Training Center, Orlando, Fl
Robert Ahlers, Project Manager

[1]Boose, J. H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition*, March, 1 (1), 3-37.

[2]Gaines, B. R. (1988). An overview of knowledge-acquisition and transfer. in *Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B. R & Boose, J. H., Eds., *Knowledge-Based Systems, Vol.1*, New York: Academic Press, 3-22.

[3]Littman, D. C. (1988). Modelling human expertise in knowledge engineering: some preliminary observations. in *Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B. R. & Boose, J. H., Eds., *Knowledge-Based Systems*, Vol.1, New York: Academic Press, 93-104.

[4]Messinger, E. B., Rowe, L. A. & Henry, R. R. (1991). A divide-and-conquer algorithm for the layout of large directed graphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21 (1), 1-11.

[5]de Kleer, J., Doyle, J., Steele, G. L., Jr. & Sussman, G. J. (1985). AMORD: Explicit Control of Reasoning. in *Readings in Knowledge Representation*, Brachman, R.

J. & Levesque, H. J., Eds., Los Altos, CA: Morgan-Kaufmann Publishers, Inc., 345-355.

[6]Loftin, R. B., Wang, L., Baffes, L. & Hua, G. (1988). An Intelligent Training System for Space Shuttle Flight Controllers. *Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence*, held May 24, 1988, at NASA/Goddard Space Flight Center, Greenbelt, Md, 3-10.

[7]Loftin, R. B., Wang, L., Baffes, P. & Hua, L. (1989). An Intelligent System for Training Space Shuttle Flight Controllers in Satellite Deployment Procedures. *Machine-Mediated Learning*, 3, 43-47.

[8]Payne, S., & Green, T. (1986). Task -action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, 2, 93-133.

[9]Wilson, M. (1989). Task models for knowledge elicitation. in *Knowledge Elicitation: Principles, Techniques and Applications*, Diaper, D., Ed., New York: Ellis Horwood, 197-219.

[10]Bylander, T. & Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: the 'right' level of abstraction for knowledge acquisition, *International Journal of Man-Machine Studies*, 26, 231-243.

[11]Wilson, M. D., Barnard, P. & MacLean, A. (1985). Analysing the learning of command sequences in a menu system. Johnson, P., & Cook, S., Eds., *People and Computers: Designing the Interface*, Cambridge: Cambridge University Press, 89-102.

[12]Akscyn, R. M., McCracken, D. L. & Yoder, E. A. (1988). "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations", *Communications of the ACM*, July, 31 (7), 820-834.

N93-11938

Mailing Address:
Post Office Box 523
Milwaukee, Wisconsin 53201-0523

Astronautics Corporation of America

An Original Paper To Be Submitted To:

Space Technology Interagency Group
Space Operations, Applications, and Research Symposium
SOAR 91 Conference at NASA / Johnson Space Center

A METHODOLOGY TO EMULATE AND EVALUATE A PRODUCTIVE VIRTUAL WORKSTATION

Jointly submitted by :

Dr. David Krubsack, Marquette University
Mr. David Haberman, Astronautics Corporation of America

Based upon work accomplished at:

The Advanced Controls Technology (ACT) Laboratory
at Marquette University and the Astronautics
Corporation of America Technology Center.

ABSTRACT FOR INTELLIGENT SYSTEMS

The Advanced Display & Computer Augmented Control (ADCACS) Program at ACT is sponsored by NASA Ames to investigate the broad field of technologies which must be combined to design a "virtual" workstation for the Space Station Freedom. This program is progressing in several areas and resulted in the definition of requirements for a workstation. A unique combination of technologies at the ACT Laboratory have been networked to effectively create an experimental environment. This experimental environment allows the integration of nonconventional input devices with a high power graphics engine within the framework of an expert system shell which coordinates the heterogenous inputs with the "virtual" presentation. The flexibility of the workstation is evolved as experiments are designed and conducted to evaluate the condition descriptions and rule sets of the expert system shell and its effectiveness in driving the graphics engine. Workstation productivity has been defined by the achievable performance in the emulator of the calibrated "sensitivity" of input devices, the graphics presentation, the possible optical enhancements to achieve a wide field of view color image and the flexibility of conditional descriptions in the expert system shell in adapting to prototype problems.

The authors propose to provide a paper and a joint presentation which would include video tape sequences color slides and overhead transparancies.

INTELLIGENT COMPUTER-ASSISTED TRAINING TESTBED FOR SPACE RELATED APPLICATION

The technologies underlying Intelligent Tutoring Systems have matured to the point that it is now possible to create authoring shells which allow non-programmers to develop and deliver them on microcomputers. However, work is necessary to discover the functions important to include in these shells and the best design for tutors built with them. This work describes one such shell called Microcomputer Intelligence for Technical Training (MITT) Writer and the development and evaluation of early space related tutors built with it.

Session I5: MISSION OPERATIONS

Session Chair: Lynne Cooper

OPERATOR ASSISTANT TO SUPPORT DEEP SPACE NETWORK LINK MONITOR & CONTROL

Lynne P. Cooper
Rajiv Desai
Elmain Martinez
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
M/S 301-490
818-354-3252

ABSTRACT

Preparing the Deep Space Network (DSN) stations to support spacecraft missions (referred to as *pre-cal,* for pre-calibration) is currently an operator and time intensive activity. Operators are responsible for sending and monitoring several hundred operator directives, messages, and warnings. Operator directives are used to configure and calibrate the various subsystems (antenna, receiver, etc.) necessary to establish a spacecraft link. Messages and warnings are issued by the subsystems upon completion of an operation, change of status, or an anomalous condition.

Some portions of pre-cal are logically parallel. Significant time savings could be realized if the existing Link Monitor and Control system (LMC) could support the operator in exploiting the parallelism inherent in pre-cal activities. Currently, operators may work on the individual subsystems in parallel, however, the burden of monitoring these parallel operations resides solely with the operator. Messages, warnings, and directives are all presented as they are received -- without being correlated to the event that triggered them.

Pre-cal is essentially an overhead activity. During pre-cal, no mission is supported, and no other activity can be performed using the equipment in the link. Therefore, it is highly desirable to reduce pre-cal time as much as possible. One approach to do this, as well as to increase efficiency and reduce errors, is the LMC Operator Assistant (OA). The LMC OA prototype demonstrates an architecture which can be used in concert with the existing LMC to exploit parallelism in pre-cal operations while providing the operators with a true monitoring capability, situational awareness[1] and positive control[2]. This paper presents an overview of the LMC OA architecture and the results from initial prototyping and test activities.

BACKGROUND

The Operator Assistant (OA), a multi-year applied research project currently in its second year, is investigating the application of Artificial Intelligence (AI) based automation techniques to ground data systems operations. The problem of introducing automation into an existing operational system is a complex one, requiring a thorough understanding of the problem domain [1]. There is an almost overwhelming temptation to attack automation piecemeal, attempting to "fix" individual problems, rather than to view the system as a whole.

Our approach to automation was to first perform an in-depth systems analysis: identify the functions performed by the current system, identify problems characteristic of the existing system, identify desired new features, and develop and evaluate various functional breakdowns (human vs. computer), and then

1. Knowledge of the true state & status of the system at all times
2. a. For all control actions, there exists a means of positively verifying that the given action occurred as desired; b. the human can, at any time, return the system to manual control.

develop an architecture capable of supporting the future system.

PROBLEM DOMAIN

NASA's Deep Space Network (DSN) is a world-wide network of large (26 to 70 meters) antennas and telecommunications equipment dedicated to the support of interplanetary spacecraft. The monitor & control systems for the network and the individual Deep Space Stations (DSSs) have evolved continually since the DSN was commissioned in the 1960's. However, the underlying architecture, which depends heavily on human operators, has changed little.

After an evaluation of DSN monitor & control (M&C), the first area chosen for application of automation technology was the DSS M&C, and particularly the Link Monitor & Control (LMC) functions.

In DSN terminology, the *Link* is the string of equipment, starting with the antenna, necessary to communicate with spacecraft. The DSN is a bent pipe which enables the spacecraft controllers to command their spacecraft and receive telemetry from the spacecraft. Each spacecraft requires a unique configuration of the DSN equipment in order to establish the link. The LMC operators are responsible for knowing which configuration is needed to support a particular spacecraft, any special test procedures or calibrations, and the sequence in which to perform the necessary actions.

During the configuration period (which can last anywhere from 45 minutes to several hours), the LMC operators are responsible for identifying, parametrizing, and sending over a hundred operator directives to several different sub-systems and subassemblies, and monitoring several hundred messages, responses and warnings to determine the health, status, and performance of the link. During this time, the station and equipment is not available to support any other activities. For some types of operations, the amount of time spent performing pre-cals (2 hours) dwarfs the actual data collection time (15 minutes). The goal of the Link Monitor & Control OA is to reduce the overhead associated with pre-calibrations and to increase operator efficiency.

The operator's job is difficult due to: 1) the lack of on-line access to procedural, schedule, and general purpose information; 2) an LMC architecture which does not allow operators to query subsystems and subassemblies for parameter values; 3) an architecture which decouples the monitor data from the display data so that there are inconsistencies between the two; 4) a keyboard-only input system which is both slow and error-inducing; 5) a directive vocabulary of over a thousand different directives without any form of standardization; and 6) an architecture which is so overladen with false-alarms that system warnings are often ignored.

Many of these deficiencies are the targets of on-going DSN upgrade activities. Therefore, the automation architecture developed for link monitor & control has to look beyond existing discrepancies, although it must take them into account for near-term implementations.

ARCHITECTURE

The LMC Operator Assistant architecture consists of five main processing modules, as depicted in Figure 1: 1) Parameter Selection; 2) Dependency Network; 3) Display Generation and Interface Management; 4) DSN Interface; and 5) Reactive Monitor, Diagnosis, and Control. Each of these modules performs specific functions which are an integral part of LMC operations. These modules are supported by several data and knowledge bases and secondary processes such as logging and report generation. The main components of the OA are discussed in the following sections.

Support Databases

As with most AI-based systems, a substantial amount of effort must be devoted to interfacing the support information and transforming it into a computer-readable and usable format [2; 3]. For the Operator Assistant, this required creating and organizing data and knowledge bases to contain the information currently used by the operations personnel. For example, the operators now

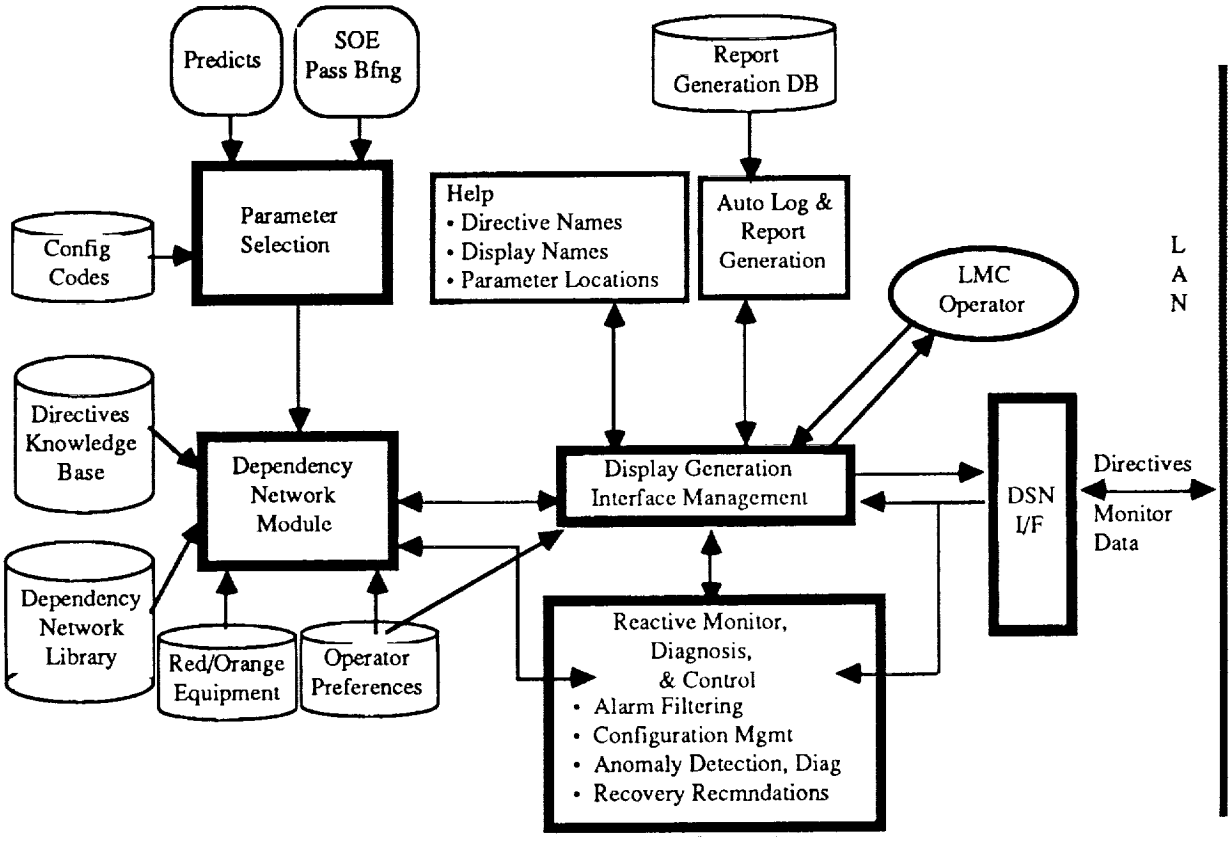


Figure 1: LMC OPERATOR ASSISTANT ARCHITECTURE

use a variety of inputs to determine exactly how they will perform a given activity. These inputs include daily schedules which identify what and when to perform activities, Sequence of Events (SOEs) which give detailed sequencing information and time critical information (e.g. *Acquire the spacecraft at time t=day::hour:min:sec*), and Predicts, which are specific predicted values necessary to communicate with the spacecraft (e.g. position, transmitter frequency). Currently, this information is available to the operator primarily in hardcopy format -- with little or no capability to edit, sort, filter, or transfer it on line. This results in the operators often having to manually enter large tables of data. Future upgrades are looking at making this information available on line. However, part of the OA architecture is to define a preferred representation format for this data.

Knowledge Bases

The key knowledge bases in the OA architecture are the Directives Knowledge Base (KB) and the Dependency Network Library. The Directives KB identifies each of the directives used to communicate with the subsystems and subassemblies. The basic information includes the directive name, description, parameters and directions for filling the parameters, expected responses, associated monitor functions, and time-out & clock information.

The individual dependency networks, stored in the Dependency Network Library, contain the information to configure, calibrate, test, and operate the link. Each network is comprised of operator directives, the post- and pre- conditions associated with a particular pass, and the predecessor and successor rela-

tionships between nodes.

Dependency Network Module

The Dependency Network Module initializes the dependency network defined in the library. First, temporal constraints are overlayed on the network resulting in a Temporal Dependency Network (TDN). The Parameter Selection Module then sets the values of directive parameters based on the most up-to-date information available. Finally, the TDN is passed to the Monitor, Diagnosis & Control Module for execution.

The contents of each node in the network include the sequence of directives needed to accomplish the node, the values of any parameters which are predetermined by the type of choice of activity, preconditions, postconditions, and predefined contingencies.

The TDN is a logical representation of the activity. It includes all of the required steps, and also identifies optional/desired steps, and sub-optimal operating conditions. It is a standardized, yet flexible representation of a DSN activity which incorporates the knowledge of the operations, engineering, and science personnel.

Reactive Monitor, Diagnosis & Control

The Reactive Monitor, Diagnosis, and Control Module is the centerpiece of the Operator Assistant architecture. It is responsible for scheduling the execution of the TDN directives, queuing them for execution, spawning the monitor and timing functions associated with each directive, collecting and distributing responses, detecting & diagnosing anomalies in TDN execution, recommending repair strategies, and replanning TDN execution when necessary.

This module was implemented using a blackboard architecture, as shown in Figure 2. The scheduler, queue manager, monitor manager, response and directive classification and disbursement, and clock manager functions have all been implemented as part of the blackboard.

The blackboard paradigm is a general-purpose architecture which can support a number of different types of applications including monitoring and diagnosis [4; 5], problem solving [6], and intelligent tutoring [7]. For the Operator Assistant, we used the blackboard not only to exchange knowledge about the system in order to support monitoring and diagnosis, but also as a mechanism for controlling the link equipment and interfacing to the DSN.

Display & Interface Management

The Display Generation Interface Management module is responsible for ensuring that the human operators see the information needed to perform their functions in the system. The goals are to 1) ensure that the human operator at all times can determine the health, status, configuration, and performance of the link, and 2) ensure that the human operator can at any time intervene in an operation in progress and return it to manual control.

DSN Interface

The DSN Interface module is the communications interface to the DSN Local Area Network (LAN) for monitor & control. This module is responsible for 1) acquiring information off the LAN, formatting it, and delivering it to the Operator Assistant, and 2) performing the inverse functions to allow the Operator Assistant to send control information across the LAN.

STATUS

The Operator Assistant Prototype, Version 1.0, was developed on a Macintosh II system, using Common LISP. The five primary modules and the portions of the supporting knowledge and databases required for the demonstration domain were implemented, integrated and tested in a laboratory setting. The diagnostician is currently being designed and will be incorporated into the next version of the OA prototype, to be released in September 1991.

The laboratory test of the Operator Assistant was accomplished using a SUN 3 computer running a subsystem response simulator which

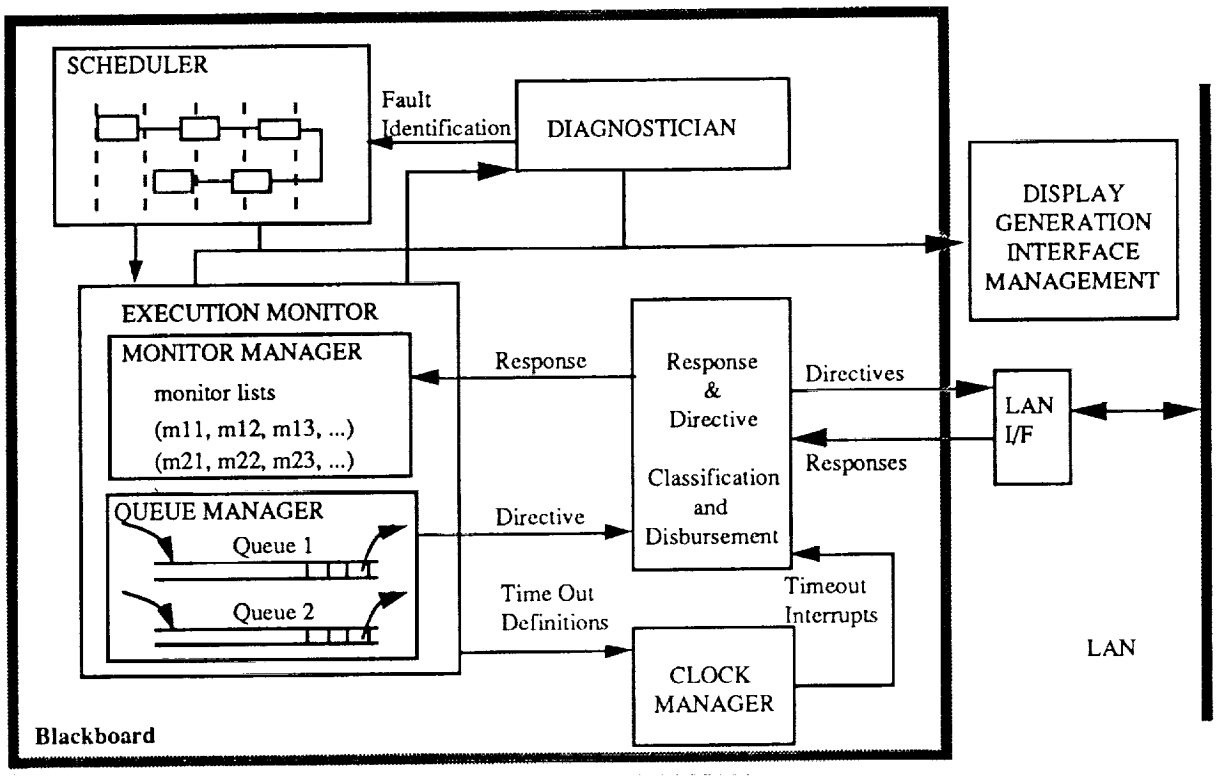


Figure 2. MONITOR, DIAGNOSIS, & CONTROL BLACKBOARD

was written in C. The Operator Assistant and Response Simulator were connected using Ethernet running TCP/IP. The tests demonstrated that the OA Prototype Architecture was appropriate and would meet system requirements. However, the test also highlighted some networking and memory management problems which will be addressed during the transition to the next version of the OA Prototype.

DEMONSTRATION DOMAIN: VLBI

The Operator Assistant was demonstrated for the Very Long Baseline Interferometry (VLBI) domain. VLBI is essentially a positioning technique which uses triangulation techniques to very accurately determine spacecraft velocity vectors. To perform a VLBI track, the DSN must configure two Deep Space Stations (located a continent apart) to the exact same configuration. Using differences in the phase and timing parameters of incoming signals, the VLBI scientists use differencing techniques to extract the desired data type.

VLBI operations are extremely difficult for operators because 1) they are not performed often, 2) they require the link equipment to be configured in a way different from standard telemetry and commanding activities, and 3) the underlying scientific theory for VLBI is difficult to understand in the context of operational options. Because of these characteristics, and the user-documented need for improving in VLBI performance through operability enhancements, the VLBI domain was chosen to demonstrate the OA.

The TDN for a VLBI[3] pass was developed. It incorporated procedural information gathered

3. Specifically, a VLBI Delta-DOR (Double Differential One-Way Ranging) pass for the Galileo spacecraft using the 70-meter antenna (DSS-14) at the Goldstone Deep Space Communications Complex. The TDN would be slightly different for other spacecraft, other antennas, or for other types of VLBI science activities.

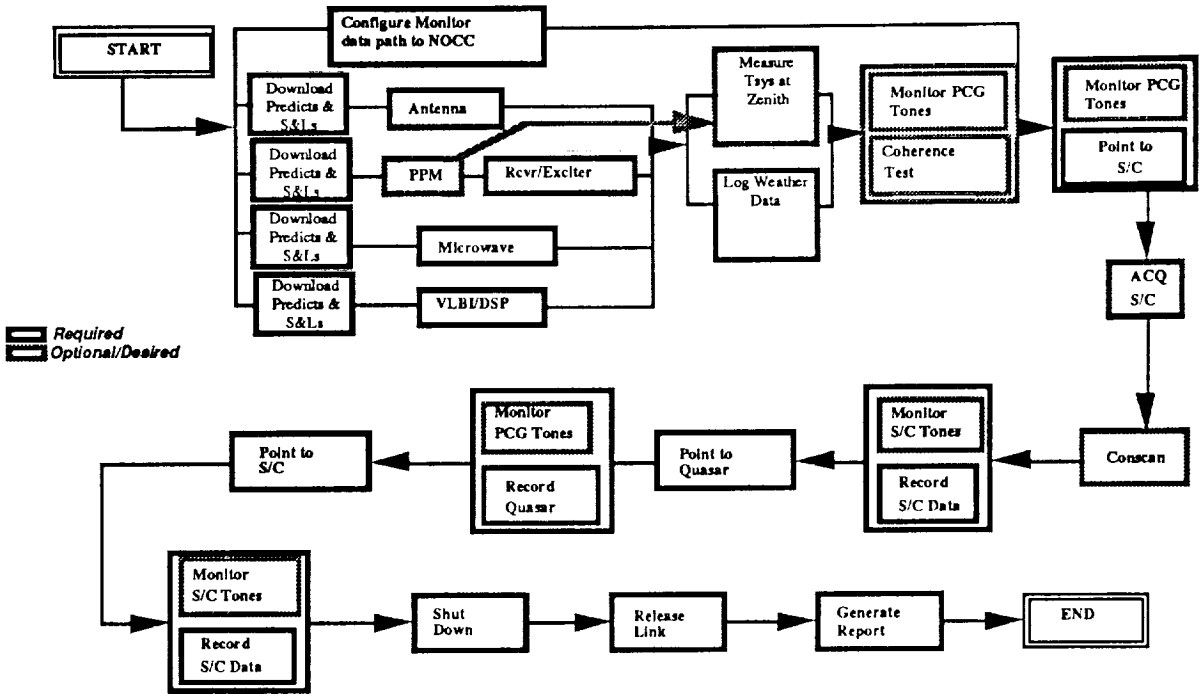


Figure 3. VLBI TEMPORAL DEPENDENCY NETWORK

from the published documentation, the link operations personnel at the Goldstone Deep Space Communications Complex, DSN systems and subsystems engineering personnel, and the VLBI scientists. The TDN shown in Figure 3 is a high-level overview of a VLBI pass. It represents the first time that all of the different components of a operational procedure were collected in the same place and the constraints, dependencies, and desired features explicitly stated.

CONCLUSION

The Operator Assistant prototype concept & architecture was the result of an in-depth analysis of how AI-based automation techniques could be incorporated into DSN operations. DSN operators, and specifically LMC operators, are part of a complex human-machine system which places a heavy burden on the human portion of the system to make it all work. The Operator Assistant is the first step in creating a new functional distribution between the operators and the systems they control and use. In this environment it is very difficult to improve monitoring without first improving the underlying control structure -- we have attacked the problem from both sides to address the issues of situational awareness and positive control. The resulting system has been demonstrated in a laboratory setting and is currently being upgraded to be demonstrated in a real-time operational environment.

ACKNOWLEDGEMENTS

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors would like to acknowledge the contributions of the following people: Sue Finley, Joe Jupin, George Resch, and Juan Urista.

REFERENCES

[1] FitzGerald, Jerry, Ardra F. FitzGerald, and Warren D. Stallings, Jr., "Understanding the Existing System," FUNDAMENTALS OF SYSTEMS ANALYSIS, Second Edition, John Wiley & Sons, New York, NY, 1981.

[2] James, Mark, and Denise Lawson, "SHARP: A Multi-Mission Artificial Intelligence System for Spacecraft Telemetry Monitoring and Diagnosis," JPL Publication 89-23, Jet Propulsion Laboratory Internal Document, May 1, 1989.

[3] Irgon, Adam, Jean Zolnowski, Karen J. Murray, and Marvin Gersho, "Expert System Development, A Retrospective View of Five Systems," IEEE EXPERT, Vol. 3, Number 3, June 1990, pp. 25 - 40.

[4] Atkinson, David, Mark James, and R. Gaius Martin, "Automated Monitoring of Spacecraft Health and Status,", SOAR Symposium, Albuquerque, NM, June 1990, pp. 244 - 258.

[5] Lesser, Victor and Daniel D. Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," BLACKBOARD SYSTEMS, ed. Robert Engelmore and Tony Morgan, Addison-Wesley Publishing Co. Menlo Park, CA, 1988, pp. 353 - 386.

[6] Pearson, G., "Mission Planning within the Framework of the Blackboard Model," BLACKBOARD SYSTEMS, ed. Robert Engelmore and Tony Morgan, Addison-Wesley Publishing Co. Menlo Park, CA, 1988, pp. 433 - 442.

[7] Murray, William, "A Blackboard-based Dynamic Instructional Planner," Proceedings of AAAI-90, Boston, MA, July - August 1990, pp. 434 - 441.

N93-11940

SPACE COMMUNICATION ARTIFICIAL INTELLIGENCE FOR LINK EVALUATION TERMINAL (SCAILET)

Anoosh K. Shahidi*
Sverdrup Technology, Inc.

Richard F. Schlegelmilch
The University of Akron

Edward J. Petrik
NASA, Lewis Research Center

Jerry L. Walters
NASA, Lewis Research Center

ABSTRACT

A software application to assist end-users of the high burst rate (HBR) link evaluation terminal (LET) for satellite communications is being developed. The HBR LET system developed at NASA Lewis Research Center is an element of the Advanced Communications Technology Satellite (ACTS) Project.

The HBR LET is divided into seven major subsystems, each with its own expert. Programming scripts, test procedures defined by design engineers, set up the HBR LET system. These programming scripts are cryptic, hard to maintain and require a steep learning curve. These scripts were developed by the system engineers who will not be available for the end-users of the system.

To increase end-user productivity a friendly interface needs to be added to the system. One possible solution is to provided the user with adequate documentation to perform the needed tasks. With the complexity of this system the vast amount of documentation needed would be overwhelming and the information would be hard to retrieve. With limited resources, maintenance is another reason for not using this form of documentation.

An advanced form of interaction is being explored using current computer techniques. This application, which incorporates a combination of multimedia and artificial intelligence (AI) techniques to provided end-users with an intelligent interface to the HBR LET system, is comprised of an intelligent assistant, intelligent tutoring, and hypermedia documentation. The intelligent assistant and tutoring systems address the critical programming needs of the end-user.

INTRODUCTION

A software application to assist end-users of the link evaluation terminal (LET) for satellite communications is being developed. This software application incorporates artificial intelligence (AI) techniques and will be deployed as an interface to LET. The high burst rate (HBR) LET provides 30 GHz transmitting/20 GHz receiving, 220/110 Mbps capability for wideband communications technology experiments with the Advanced Communications Technology Satellite (ACTS). The HBR LET can monitor and evaluate the integrity of the HBR communications uplink and downlink to the ACTS satellite. The uplink HBR transmission is performed by bursting the bit-pattern as a modulated signal to the satellite. The HBR LET can determine the bit error rate (BER) under various atmospheric conditions by comparing the transmitted bit pattern with the received bit pattern. An algorithm for power augmentation will be applied to enhance the system's BER performance at reduced signal strength caused by adverse conditions.

The HBR LET terminal consists of seven major subsystems:

- Antenna subsystem
- Radio frequency (RF) transmitter subsystem
- RF receiver subsystem
- Control and performance monitor (C&PM) computer subsystem
- Local loopback subsystem at RF
- Modulation and BER measurements subsystem
- Calibration subsystem

The C&PM computer controls and monitors all the other subsystems through an IEEE488 interface. HBR LET experiments with the ACTS satellite will be initiated by users through the C&PM experiment control and monitor (ECM) software. The ECM software was developed on a Concurrent 3205 minicomputer in FORTRAN, which provides the end-user with the following capabilities:

- Individual instrument control
- Interactive interface used to communicate with the digital ground terminal
- Ability to conduct BER measurements
- User-controlled data acquisition

Programming scripts, defined by the design engineer, set up the HBR LET terminal by programming subsystem devices through IEEE488 interfaces. However, the scripts are difficult to use, require a steep learning curve, are cryptic, and are hard to maintain. The combination of the learning curve and the complexities involved with editing the script files may discourage end-users from utilizing the full capabilities of the HBR LET system.

The following SCAILET features will improve the HBR LET system and enhance the end-user's ability to perform the experiments:

INTELLIGENT ASSISTANT

An intelligent assistant is a software program that will aid the user in operating of the HBR LET components. Friendly human interfaces shield the user from the script and complexities of the HBR LET system and furthermore aid performing iterative setup tasks. Any intelligent assistant also contains sufficient information about the HBR LET system to alert the user to erroneous actions.

The intelligent assistant uses a personal computer to provide a dynamic schematic diagram of the overall system . The schematic diagram provides a graphic user interface which serves as a "front-end" to the HBR LET system. The user can control the system by interacting with the schematic diagram. An expert system handles requested changes in the system. The changes will then be reflected dynamically on the schematic diagram.

The dynamic graphics are implemented using the Choreographer Graphical User Interface toolkit, developed by GUIdance Technologies Corp. The expert system shell used for this project is KAPPA PC developed by Intellicorp Corp. KAPPA PC is a hybrid expert system shell which has both a complete object oriented programming system and a rule based expert system. All HBR LET devices are implemented in KAPPA PC objects. The devices are connected to one another using message passing to create a dynamic model of the system. Rules are used to create programming scripts.

MULTIMEDIA DOCUMENTATION

Multimedia is a software package that applies a combination of text, graphics, voice, and video technologies in a computer application tool. The combination of these technologies provides a more powerful tool than any one medium alone. Text and graphics information can then be combined within the application. Similarly, associated voice and video information can be integrated into the application. This module is implemented using the ToolBook hypertext shell developed by Asymetrix Corp.

The documentation for the HBR LET subsystems is being developed by different design engineers. However, the end-user will need to see the association among the subsystems. Multimedia allows the user to look at a graphic image of a schematic diagram, and request written specifications of the components, more detailed images, or actual assembly diagrams. Multimedia allows the users to follow schematics' links and visit subsystems within the circuitry. Using printed documents would be less friendly and require much more time and effort to understand. This module is connected to the intelligent assistant so the user can access relevant documentation anytime during system programming.

FUTURE DIRECTION: INTELLIGENT TUTOR

Computer assisted instruction (CAI) is a traditional computer based training program which takes the user through a predetermined set of lessons. The advent of artificial intelligence technology and advances in cognitive psychology gave rise to intelligent tutoring systems (ITS) as an improvement to CAI. In an ITS environment, the curriculum designer determines what concepts the student should learn in a lesson. The student is then taken through subjects to see which concepts he/she is lacking. The program then determines the curriculum depending on the needs of the student.

For HBR LET, an initial overview of individual system components is necessary to aid in understanding the complete system. Concepts important to the operation of each HBR LET subsystem will be identified for SCAILET after the multimedia documentation and intelligent assistant tasks are completed . A guided learning process, which incorporates the use of a simulator, will be developed to provide ITS instruction on the operation of the HBR LET system.

ACKNOWLEDGMENTS

This project is being developed at and funded by the Space Electronics Division of the NASA Lewis Research Center. We would like to thank Mr. Rich Rienhart of Analax Corp., the programmer of the ECM software, for his excellent work and cooperation with the SCAILET team.We would also like to thank the NASA Lewis Research Center ACTS Project Office.

*Principle contact: Anoosh K. Shahidi, Sverdrup Technology Inc. Mail Stop SVR-1, 2001 Aerospace Parkway, Brook Park, Ohio 44142 (216) 891-2213

BIBLIOGRAPHY

1. "Using Open Script: ToolBook Software Construction Set for Windows". Asymetrix Corp., Bellevue, Washington 98004. (1990)

2. "KAPPA PC Reference Manual". Intellicorp Corp., Mountain View, CA (1989).

3. " Choreographer Reference Manual". Guidance Technologies Corp. (1989)

Advanced Satellite Workstation

An Integrated Workstation Environment For Operational Support of Satellite System Planning & Analysis

Stewart A. Sutton

The Aerospace Corporation
Los Angeles, California

Abstract

This paper describes a prototype integrated environment, the Advanced Satellite Workstation (ASW), that has been developed and delivered for evaluation and operator feedback in an operational satellite control center. The current ASW hardware consists of a Sun Workstation and Macintosh II Workstation connected via an ethernet Network Hardware and Software, Laser Disk System, Optical Storage System, and Telemetry Data File Interface. The central mission of ASW is to provide an intelligent decision support and training environment for operator/analysts of complex systems such as satellites. There have been many workstation implementations recently which incorporate graphical telemetry displays and expert systems. ASW is a considerably broader look at intelligent, integrated environments for decision support, based upon the premise that the central features of such an environment are intelligent data access and integrated toolsets. A variety of tools have been constructed in support of this prototype environment including: an automated pass planner for scheduling vehicle support activities, architectural modeler for hierarchical simulation and analysis of satellite vehicle subsystems, multimedia-based information systems that provide an intuitive and easily accessible interface to Orbit Operations Handbooks and other relevant support documentation, and a data analysis architecture that integrates user modifiable telemetry display systems, expert systems for background data analysis, and interfaces to the multimedia system via inter-process communication.

Advanced Satellite Workstation

An Integrated Workstation Environment For Operational Support of Satellite System Planning & Analysis

Stewart A. Sutton

The Aerospace Corporation
Los Angeles, California

Abstract

This paper describes a prototype integrated environment, the Advanced Satellite Workstation (ASW), that has been developed and delivered for evaluation and operator feedback in an operational satellite control center. The current ASW hardware consists of a Sun Workstation and Macintosh II Workstation connected via an ethernet Network Hardware and Software, Laser Disk System, Optical Storage System, and Telemetry Data File Interface. The central mission of ASW is to provide an intelligent decision support and training environment for operator/analysts of complex systems such as satellites. There have been many workstation implementations recently which incorporate graphical telemetry displays and expert systems. ASW is a considerably broader look at intelligent, integrated environments for decision support, based upon the premise that the central features of such an environment are intelligent data access and integrated toolsets. A variety of tools have been constructed in support of this prototype environment including: an automated pass planner for scheduling vehicle support activities, architectural modeler for hierarchical simulation and analysis of satellite vehicle subsystems, multimedia-based information systems that provide an intuitive and easily accessible interface to Orbit Operations Handbooks and other relevant support documentation, and a data analysis architecture that integrates user modifiable telemetry display systems, expert systems for background data analysis, and interfaces to the multimedia system via inter-process communication.

Executive Summary

The Advanced Satellite Workstation (ASW) project was conceived as a mechanism for demonstration of advanced information technology as applied to satellite support activities. Initial prototypes concentrated on providing graphical telemetry displays ("electronic strip-charts") and very specific expert system modules for anomaly diagnosis. As experienced was gained with the technology, and the technology itself grew in capability, it became apparent that the definition of an intelligent environment for decision support systems such as a satellite workstation encompasses more than expert systems and displays. Central to the evolving ASW concept is a robust multimedia data architecture and an integrated, communicating set of tools. Interation with this ASW prototype system is providing a method for exploring operational questions relating to satellite support activities.

ASW has recently been installed in an active Mission Control Complex. The current hardware is composed of Sun and Macintosh II workstations. These two workstations are connected via ethernet, and can interface to data products within the operational environment over this network. The technologies being applied are expert systems, graphical user interfaces, graphical data visualization, hypermedia and multimedia information systems, and hierarchical design and modeling environments. The specific applications that are being targeted as part of the current ASW implementation are: Automated Pass Plan Generator, Electronic Orbit Operations Handbook, Telemetry Data Graphical Display System, Telemetry Data Trend Analysis System, Vehicle Subsystem Hierarchical Modeler, and On-orbit Event Scheduler. Prototypes of the first four have been integrated and delivered. Major emphasis is on communication and data sharing among these applications to provide an integrated environment for decision support.

Introduction

ASW Project History

The Advanced Satellite Workstation project was initiated in 1985. The primary initial focus of the project was the technology transfer of expert systems to satellite support activities. By mid-'85 an prototype system was completed on a Symbolics computer. This first prototype was designed to solve anomalies in the attitude control system of the DSCS III spacecraft. The initial system was useful in demonstrating the utility of rule-based systems as applied to spacecraft system anomaly resolution.

Further work on this project resulted in the development of a lisp-based satellite architecture browser. The browser was conceived as a tool that would allow engineers to enter (in a hierarchical manner) schematics and subsystem documentation related to the space vehicle. This information could be browsed by the user in a structured fashion, and provide insight into any component of the satellite subsystem to the finest level of detail installed in the browser. By 1986 the satellite architecture browser incorporated diagnosis and simple simulation capabilities. This provided the user with the ability to simulate the operation of the hierarchical models installed into the browser; while comparing the simulated output with actual telemetry data from the satellite vehicle.

1986 also saw the development of an expert system for anomaly resolution in the GPS spacecraft. Again, the attitude control system was the focus of the effort. Specifically, the system did long term trend analysis of the telemetry data representing the spin-rate of the reaction control wheels. Upon analysis of this data, the expert system could determine if upon exit from an eclipse the vehicle would tend to exhibit excessive roll and pitch. Recommendations from the expert system would alert the operators to the condition of the vehicle prior to exit from eclipse so that the vehicle attitude could be closely monitored. This system demonstrated the value of expert systems for long-term, automated monitoring of telemetry data not practical for human operators.

In 1987 development was initiated on hypermedia-based information system that would provide the satellite operators with access to documentation. This system was also used as a tool to rapidly develop and test (in conjunction with satellite operators) different user displays and interfaces.

The concept of an integrated architecture that would combine the electronic documentation system with the expert system based analyst, selectable telemetry displays, and the architecture modeler was developed in 1987. By 1988 the entire ASW prototype was migrated from the Symbolics lisp processor into a general purpose workstation. By moving to a more general purpose workstation the project would be able to target delivery into operational programs. 1989 efforts focused on the integration of general purpose tools within the evolving environment that comprised the ASW satellite analyst's workbench. In 1990 the first operational prototype system was fielded and is currently being tested by satellite operators.

Problem

Mission planning, analysis, and command and control functions for military systems have increased significantly in complexity and can be expected to become still more demanding as systems themselves become larger, more complex, and more highly automated. This trend is especially true for large space systems. Future Air Force satellites may have tens of processors onboard, with increased onboard autonomy and higher telemetry and payload data rates. The number of vehicles in a satellite constellation is also increasing. At the same time, budget pressures and shifting of operations from the development community to the user community means fewer, less well trained operators and analysts. To deal with this set of circumstances, we must use new ground station technology to provide operators with a better environment. Intelligent tools for decision support and training can provide significant leverage both in terms of functionality and user friendly support to less experienced operators. The following technical areas are being explored as part of the ASW prototype activity:

Technical Area	Questions Being Explored
Expert Systems	*What areas within satellite support activities are best suited towards implementation using expert systems? Can experience on a specific satellite subsystem for one program improve the efficiency of a similar expert system development for a different program?*
Telemetry Processing and Display Systems	*Efficiency of user configurable telemetry display systems. Graphic display of telemetry versus numeric display of measurands. Local processing of telemetry data as needed versus batch processing. User interface and control mechanisms.*
Hypermedia and Multimedia Systems	*Can an interface to satellite vehicle documentation be developed that is intuitive for all users of the system? Does having random access to as-built photographs of the satellite vehicle assist the operator in normal operations or during anomaly resolution procedures? How should multimedic data be structured to provide scalable systems with efficient user and programmed data access and query capabilities? What procedures will streamline the development of electronic multimedia data systems?*
Modeling and Visualization	*What modeling and simulation techniques can be used most effectively in an operational environment? How can advanced visualization techniques be used in conjunction with modeling to improve operator understanding?*

Artificial Neural Networks and Fuzzy Logic Systems

By developing system models of space vehicle subsystems using techniques that are based on examples of real world data, can a decision support system be developed that provides good recommendations under conditions that can include incomplete or inconsistent data?

Solution

A significant number of projects to address the above issues are underway at Space Systems Division, government labs, and contractors, focusing on workstation environments for satellite commanding and control. Graphical telemetry displays and simple use of expert systems for anomaly resolution have been demonstrated in support of satellite programs, including DSCS, DMSP, IUS, GPS, and many others. Expert systems are being used operationally by NASA at JSC and by JPL in support of current satellites. Based upon our early prototypes, in ASW we are extending this expert system and display metaphor to include a number of technologies such as fuzzy logic, advanced visualization techniques, simulation, and multimedia/hypermedia. While each of these technologies is a powerful approach to specific tasks, the integration of these technologies in a unified support environment provides a new paradigm for operations support.

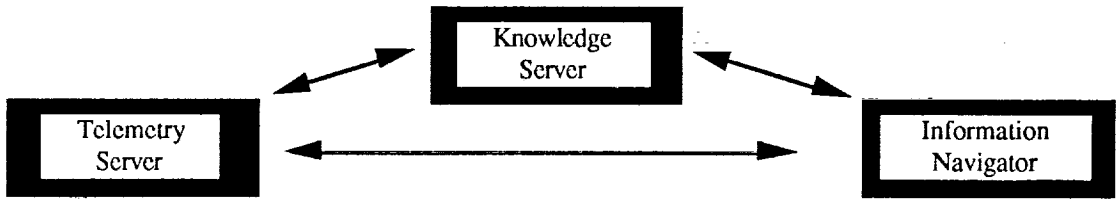
Some space vehicle anomalies can be avoided if accurate and persistent trend analysis of telemetry data is performed. ASW uses expert systems to encode the satellite subsystems' operational structure. The rule-based system makes use of objects for modeling elements of the subsystem such as batteries, high-power amplifiers, solar arrays, reaction control wheels, and thermocouples. This object-based representation of components works well for system components that can be defined by a set of equations or transfer functions. When a system component is best described by examples of its operation under varied conditions, then mechanisms such as artificial neural network models can be used to construct an object model. Object-based models of system components are integrated into the rule-based system and together these elements form the knowledge processing system for a specific spacecraft subsystem. These knowledge processing systems interface to telemetry server agents that provide data required for trend analysis and health and status processing. All of these activities occur in background as multiple processes and provide high-level messages to a user modifiable graphics display system for operator interpretation. In addition to the messages that are provided to the operator, an on-line documentation system is integrated into the architecture to provide ready access to vehicle-specific data. This documentation system can be automatically oriented to appropriate "chapters" according to the context of the analysis being performed with the workstation.

System Architecture

The system architecture of the Advanced Satellite Workstation is composed of three principal components. These components (shown in Figure-1) are the Knowledge Server, Information Navigator, and Telemetry Server. The implementation of these components and their integration with each other is a continually evolving process. In addition, ASW includes a modeling and visualization system and more specific tools for satellite pass planning, on-orbit event scheduling, etc. A fourth important component currently being defined is an object-oriented data server to provide uniform access to multimedia information.

Figure-1.

ASW System Architecture



Telemetry Server

The present telemetry server provides buffered telemetry data both for user displays and for use by the expert system. Since the current system is designed for post-pass analysis, a telemetry data file is created in shared memory with pointers to individual parameter streams. The server also controls the user interface, handling input events and using toolbox calls to provide selectable telemetry displays. The telemetry server forks a Unix process which activates the expert system, and passes pointers to shared memory to give the expert system access to telemetry. The expert system uses Unix pipes to send diagnostic messages to the telemetry server for

display to the operator. Modifications to this architecture are currently being designed to provide control of digital real-time telemetry streams provided by a dedicated preprocessor interfaced to wideband analog satellite data.

Information Navigator

The Information Navigator's primary function is presenting information to the satellite operator in an intuitive fashion. A vast amount of supporting documentation accompanies most satellite programs. Developing familiarity with the satellite program through accessing this data is an inefficient, labor intensive process. The supporting documentation that can be of particular importance during anomaly resolution procedures is often difficult to locate, and may be in the archives of the satellite vehicle's prime contractor or associated subcontractors. Accessing this information would pose a considerable challenge if it were even delivered as part of the contract, and chances are that if delivered in its most popular form (paper) it would probably find a nice dark out-of-the-way storeroom in which to reside.

Simply moving this information into electronic form is not sufficient. Standard databases are powerful tools for structured data and provide flexible query capabilities, but do not provide adequate methods for dealing with implicitly related information. The data of potential interest to the satellite operator is not just alphanumeric data, but includes engineering schematics, program schedules, animated models, and video. Object-oriented databases offer the promise of dealing with such multimedia data, but are not yet a mature technology. ASW currently deals with these different forms of information using an Information Navigator able to provide automated access to video and still data on laser disk as well as graphical and text data stored on optical and magnetic media. The Information Navigator permits hypermedia techniques to be employed to link data explicitly for user-directed browsing, and this approach has been used to develop an online Orbital Operations Handbook (OOH) for a satellite program (See Appendix-A for sample screens of the Information Navigator OOH). The Information Navigator also provides the capability to perform text searches to locate implicitly related data without explicit predefined links. The Information Navigator is currently based on the Macintosh workstation, and can communicate with the processes on the Sun via a defined 2-way communication protocol. This architecture permits both user and expert system control of the information display, as well as the passing of information from the electronic database to processes running on the Sun workstation.

Knowledge Server

In the context of ASW, a Knowledge Server encapsulates domain-specific knowledge and uses this knowledge to provide advisory messages, suggest actions, and direct the operator to documentation. Initial implementations have been limited to expert systems for specific satellite subsystems such as electrical power. In a fully operational system, multiple concurrent knowledge agents will independently process telemetry related to separate subsystems, under the control of an executive process. In ASW, expert system processes run in the background, reading telemetry from the shared memory buffer created by the Telemetry Server. These processes use Unix pipes to send messages to the operator's screen. Since messages may be voluminous under certain conditions, these message are read in a scrollable window by the operator. The Telemetry Server-Knowledge Server interface permits to operator to automatically jump to telemetry data related to messages by simply mousing on a button. At the operator's discretion, the Knowledge Server may also signal the Information Navigator to provide automatic access to documentation relevant to a current area of concern.

A more robust system could use standard expert systems for the capture of high-level knowledge and heuristics regarding system operation, while drawing on additional techniques such a fuzzy logic for approximate reasoning and dealing with continuous data with inexact boundaries. Real-world events can often be grouped into a continuous spectrum of values that are termed fuzzy sets. Fuzzy inferencing on such data can provide robustness to uncertainty and "common-sense" reasoning.

Artificial Neural Networks (ANN) can provide another technique for extending the capability of intelligent systems such as a knowledge server process. While system models can often be defined using a set of equations that precisely describe the behavior of the system, at times it is not practical to model behavior based on a set of equations. In such cases the application of an artificial neural network can provide a model that behaves like the real subsystem, provided the ANN model is developed using data from real world systems. An example of such a system would be an ANN model of a spacecraft battery. Given specific inputs reflecting the state of the space environment, loading from the spacecraft subsystems, and past charge/discharge profiles, an ANN battery model could provide predictions of power system performance. These predictions would be based on a model that was developed with real-world data. An expert system making assessment of the state of spacecraft subsystems could use such models for hypothesis exploration.

Additional Capabilities

A realistic satellite support environment must provide many additional capabilities, and these capabilities should be integrated into a consistent overall environment. At present, ASW has developed tools for automated pass planning support, mission scheduling, and systems modeling and browsing.

Pass Planner

The pass planning capability was specifically requested by operators to automate the repetitive, labor intensive process of creating detailed pass plans used to control satellite commanding and data recording during real time contacts. This tool uses many of the same toolbox calls as the telemetry server display process to provide a consistent interactive user interface. Based on input data the pass planner interrogates data files to create a consistent plan. A specialized planning language was developed to provide flexible programmatic control of this process so that it can be generalized to multiple satellite families.

Satellite Architecture Modeler

Another important tool is the Satellite Architecture Modeler (SAM). SAM is written in Smalltalk™, and runs on both the Sun and Macintosh computers. It provides a way to graphically display color diagrams representing the functional structure of a system, identifying the modules, interfaces, and communication routes. The user can navigate through the system by pointing at the object he wants to move to, and the display then changes to the part of the system centered on that object. In addition, the Modeler provides a discrete-event simulation capability. The simulation of the entire system is determined from the simulation of each fundamental component in the hierarchical structure, organized and coordinated as specified in the diagrams. The simulation of a fundamental component is specified purely locally, without reference to external entities, and the structure of the system joins these independent parts into a cooperative whole. The simulation capability was recently extended to include time, allowing feedback loops and time-varying behavior: this has greatly increased the range of possible simulations. Work is underway to fully integrate SAM with ASW.

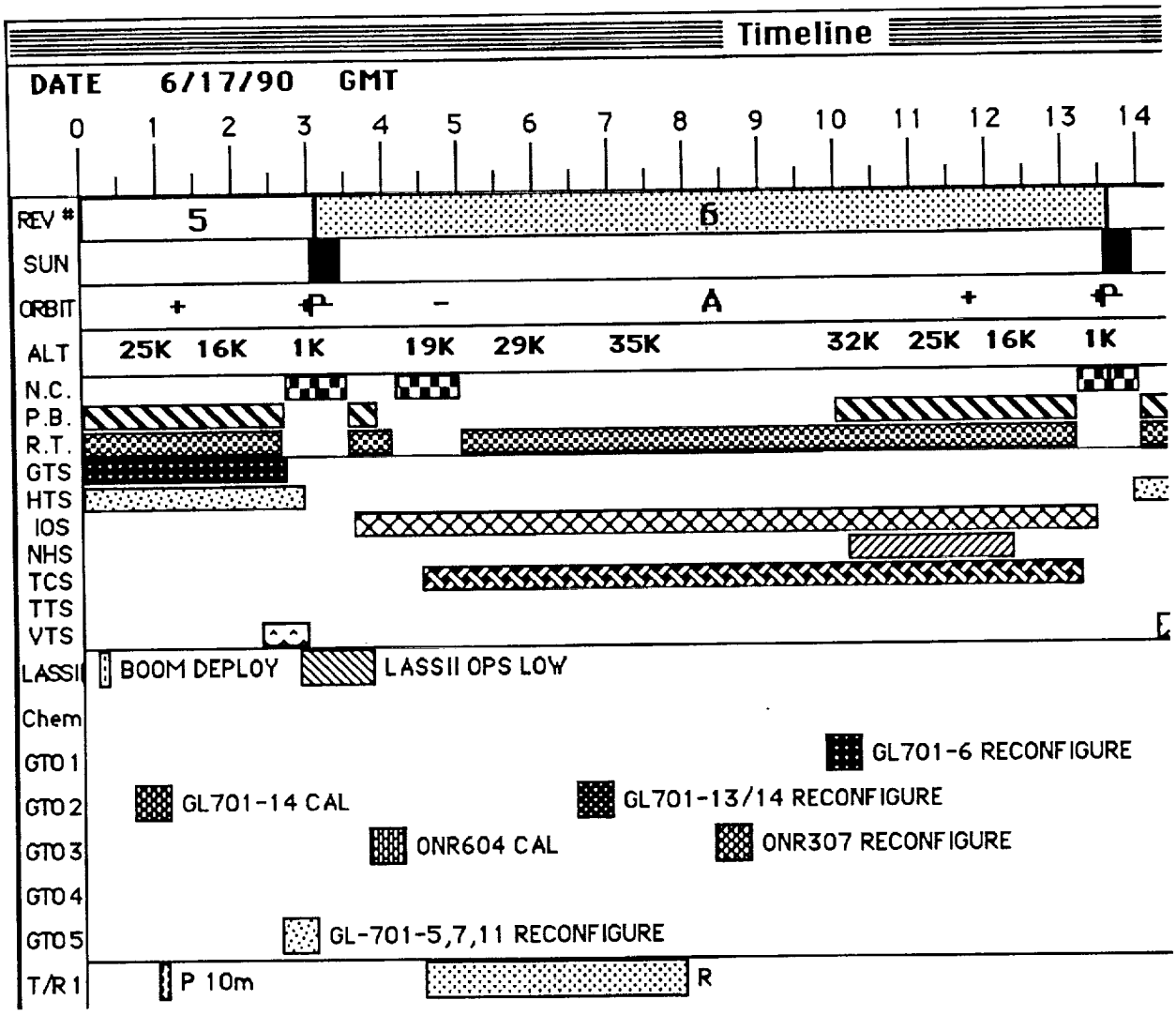
In addition, future versions of SAM will provide advanced visualization capabilities. Current space systems are deeply structured, complex mechanisms. Dealing with that complexity is straining the capacity of even trained and experienced personnel, while there is a high learning curve for new spacecraft operators. Simulation and modeling, combined with graphical output showing the unfolding development of the system being simulated, can be of significant benefit in understanding complex systems, as evidenced by the growing interest in scientific visualization. Viewing the time-varying behavior of a simulation conveys far more information to a user than static displays. For clarity, these displays must be composed according to a consistent metaphor, focusing on a particular subset of the system's behavior. This subset could be qualified by structural, functional, or dynamic considerations. There will often be multiple valid views of a system, which may range from small to large, simple to intricate, static to dynamic, independent to completely dependent. Multiple metaphors are necessary to provide appropriate views to heterogeneous subsystems. Links to neighbor systems must enable rapid browsing through associated concepts with possibly widely varying views. The user must be able to expand or contract his focus of interest, or to easily redirect his inquiries to related subjects. Since so much of the character of a system is shown through its dynamic behavior, the user must be able to easily examine and change the state of the simulation, using "what-if" strategies to explore alternative behaviors.

Timeliner

The *TIMELINER* program is a prototype stand-alone application developed in SuperCard™ for the Macintosh. It is a graphically-based scheduling tool for the development and manipulation of event timelines that correspond to events that must be scheduled onboard the space vehicle. The current version of the *TIMELINER* program was tuned towards a specific satellite program, but future extension of this work will address a more general tool for spacecraft event scheduling and management. This tools is included as part of the ASW tool set for satellite support and is hosted in the Apple Macintosh workstation.

TIMELINER has three parts: the *TIMELINER* application, the timelines, and the activity cards. Activities are defined when the user enters information pertaining to an activity on an Activity Card. These activity cards are stored in an Activity file which the user creates during his or her first session with the program. Timelines are generated by the user by choosing a time scale and scheduling activities from the Activity file. Timelines are stored in a Timeline file which is also created by the user during the first session with *TIMELINER*. A snapshot of a *TIMELINER* screen is shown in Figure-2.

Figure-2.



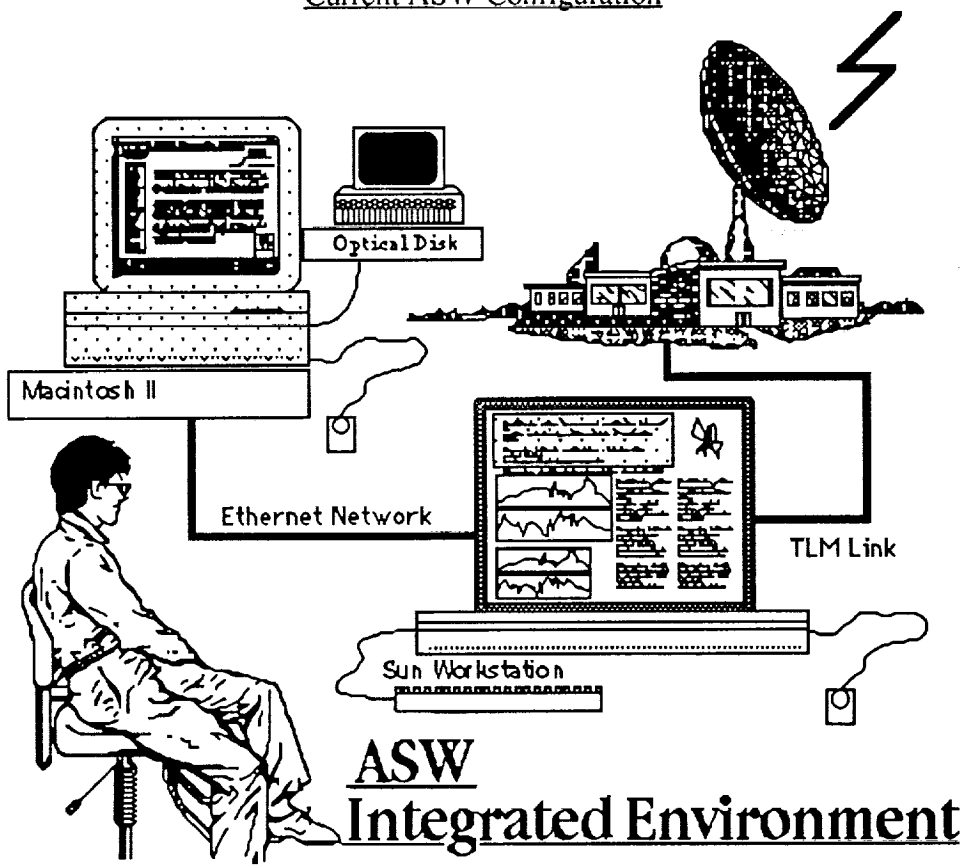
Implementation

ASW has been implemented using a Sun 3/470 workstation, Apple Macintosh IIcx workstation, laser disk player, and associated ethernet network hardware. The basic configuration is shown in Figure-3. The operator sits in front of these two workstations that are located side-by-side. The Sun workstation provides the operators primary interface to telemetry data, and data analysis tools. The Telemetry Server is implemented in the C language, using DataViews™ toolbox calls for input handling and graphics displays. In addition, the expert system processes are mostly resident on the Sun workstation. Expert systems are currently implemented using the NEXPERT™ Object expert system shell with C extensions. The Macintosh workstation provides the operator access to hypermedia-based (electronic) documentation that includes schematics, text, sound, and animation. The more exotic forms of information (sound and full-motion random accessed video) are made possible through the use of high-density optical disc, CD ROM, and video disc equipment that interfaces to the Macintosh workstation.

The Macintosh workstation and Sun workstation are connected to an ethernet network. Custom protocols based on TCP/IP are used for communication between the Sun and Macintosh computers. This network serves as the primary conduit for transfer of information between each workstation. The telemetry data is currently provided to the Sun workstation via a file system interface with a DEC PDP minicomputer. This pre-processed data is analyzed and displayed on the Sun workstation. In the future, telemetry data may be accepted over the ethernet network in *real time* from these front-end systems. The Macintosh workstation is connected to the Sun workstation via ethernet and is automatically oriented to the proper *documentation screen* according to the context of the analysis being

performed on the Sun workstation. The user may at any time direct his attention to the Macintosh and via the Information Navigator and explore the data items (text, graphics, and animation) in more detail through a hierarchical browser.

Figure-3.
Current ASW Configuration



The Orbit Operations Handbook was prototyped using HyperCard™ for the multimedia user interface environment. MacroMind Director™ was used to develop animations that play through the HyperCard environment. Studio/8™ was used to develop color images and to retouch scanned images taken from actual documentation. OmniPage™ was used as the OCR (Optical Character Recognition) package for translating printed text into ASCII files for installation into HyperCard-based text containers (fields). Swivel 3D and Super3D were used to build simple 3D models of the spacecraft and associated subsystem hardware. These 3D models were used to generate different views of the space vehicle. These 3D views were assembled in a HyperCard Stack for manipulation and orientation by the user. Later, higher quality 3D renderings were taken from videotape after being produced on a Silicon Graphics IRIS workstation. These individual views of the spacecraft were captured using Mass Micro's ColorSpace II, and ColorSpace FX video boards. The following diagrams included in Appendix-A illustrate the style of information displayed by the Information Navigator OOH (IN-OOH).

Acknowledgements

The authors would like to acknowledge Thomas E. Bleier of The Satellite Control Network Directorate for continuing support of the development of the ASW architecture from its inception in 1985 to the present. Tom Bleier has kept our perspective focused and continues to provide valuable technical and programmatic assistance. Additional acknowledgements are extended to Pinfun Tsai of The Space Test Department for supporting the development of prototype electronic OOH as well as support of TIMELINER, telemetry display systems, and program specific expert systems.

The following individuals are to be acknowledged for their dedication and long hours in ASW prototype development. Peter Homeier has provided the longest running support to this project and is the principal designer of the modeling and visualization

methods used in ASW. Grisel Hlavaty is the developer of the TIMELINER event scheduling support tool for graphical visualization and manipulation of on-orbit events. Thach Le is a co-developer of the expert system architecture as well as principal developer of the initial interface on the Sun workstation. Robert Statsinger is the designer and developer of recent subsystem specific expert systems for ASW as well as a co-architect of the shared memory architecture of the ASW workstation. Candice Yu has participated in the design, development and performed a substantial portion of the implementation of the Information Navigator Orbit Operations Handbook for ASW. Scott Zechiel has been the principal designer and developer of the telemetry display system, and workstation user interface that integrates the expert system environment, hypermedia environment, graphical display environment, and general tools environment. In addition Mr. Zechiel has been a designer and developer for the workstation interface to the electronic OOH as well as principal developer of the automated pass planner system.

N93-11942

Mission Control Center Enhancement Opportunities in the 1990s

Wayne Hartman
Artificial Intelligence Group
Loral Space & Range Systems
Sunnyvale, California

ABSTRACT

The purpose of this paper is to present a framework for understanding the major enhancement opportunities for Air Force Mission Control Centers/Test Support Centers (MCCs/TSCs) in the 1990s. Much of this paper is based on the findings of Study 232 and work currently underway in Study 2-6 for the Air Force Systems Command, Space Systems Division, Network Program Office. In this paper, we will address MCC/TSC enhancement needs primarily from the operator perspective, in terms of the increased capabilities required to improve space operations task performance.

INTRODUCTION

Study 232, "Enhancing MCC Operations Using Automation and Expert Systems Technology," was commissioned as a first step in defining the next generation of enhancements required to support MCC operations activities. Advances in technology (especially in the areas of telemetry servers, workstations, displays, graphical user interfaces, databases, hypermedia, and expert systems) provide many new opportunities for building systems and capabilities that can greatly enhance the ability of operations personnel to accomplish their assigned space operations tasks and activities.

The following sections address major operations functions and activities; current difficulties, limitations, and challenges; MCC enhancement opportunities; architecture constraints; and a framework for an Integrated Space Operations Support Environment (ISOSE).

MAJOR OPERATIONS FUNCTIONS AND ACTIVITIES

Time-, resource-, and experience-intensive tasks span the full spectrum of MCC operations functions and activities from: 1) requirements analysis, 2) to planning, 3) to scheduling, 4) to contact support, 5) to post-pass analysis (see Figure 1).

1. **Requirements analysis** involves consolidating program needs from the System Program Office, mission needs of various users, and vehicle needs from operations handbooks and directives as well as technical analyst inputs.

2. **Planning** involves translation of mission needs into planned mission activities, vehicle activity planning to translate consolidated needs into vehicle activities, and contact support plan generation to define what specific actions will be accomplished during each vehicle contact.

3. **Scheduling** involves identifying and requesting the resources necessary to accomplish MCC activities. Three levels of resources are involved. Internal MCC scheduling deals with the resources under direct control of the MCC. Personnel scheduling ensures that adequate personnel are available to accomplish scheduled operations activities. Air Force Satellite Control Network (AFSCN) scheduling deals with obtaining needed common user resources including communication links and remote tracking stations.

4. **Contact** activities include execution of contact support plans, tracking, commanding, telemetry monitoring, status monitoring, real-time telemetry analysis, contingency identification, and contingency plan execution. These occur in real time while the ground system is in contact with the space vehicle.

5. **Analysis** activities include state assessment, review of contact activities, telemetry analysis and trending, orbit analysis and determination, attitude determination, anomaly identification, anomaly analysis, and anomaly resolution. These activities are typically performed off-line.

BASIC NATURE OF OPERATIONS

MCC operations functions and activities are performed around-the-clock by military crews, primarily at Falcon Air Force Base (AFB). TSC operations activities are performed around-the-clock by contractor operations teams, primarily at Onizuka AFB. The major difference between the two is the Research and Development (R&D) emphasis and the accompanying greater need for engineering support for programs operated from the TSC. Both MCCs and TSCs require use of common communications and tracking station equipment that are a part of the AFSCN to contact and track their assigned vehicles.

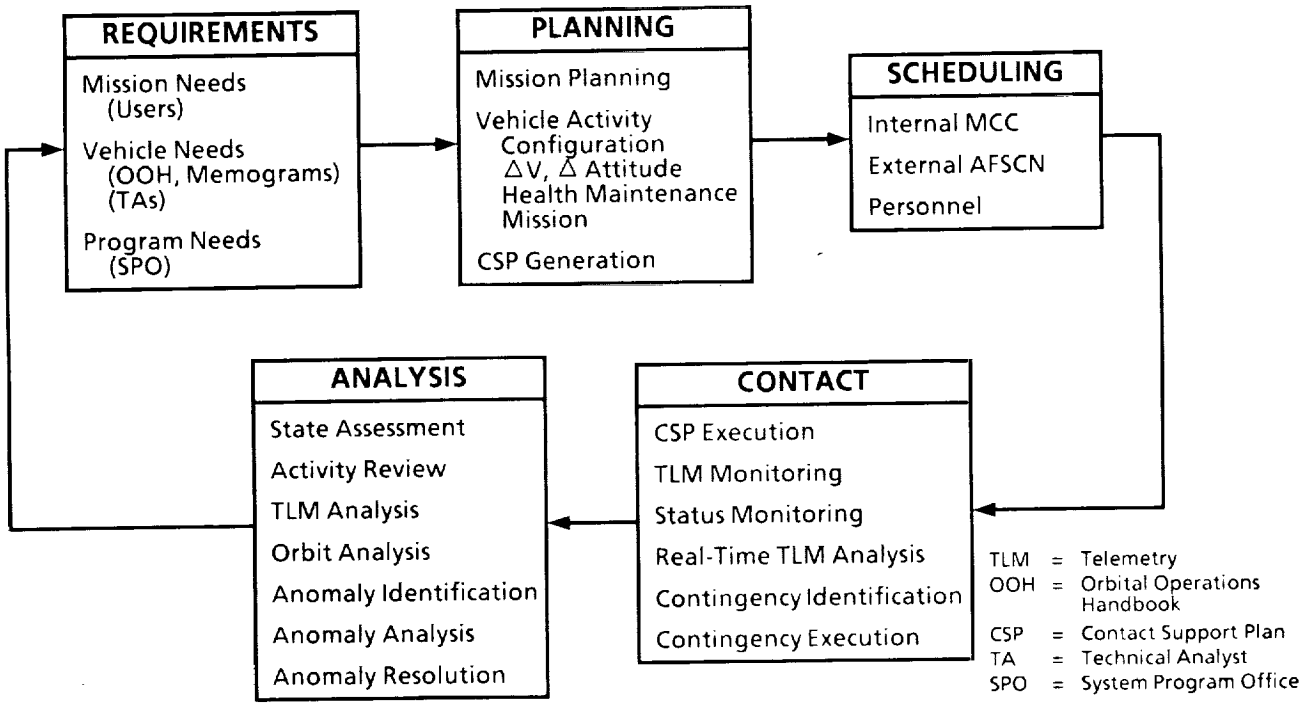


Figure 1. Major Operations Functions/Activities

NEED FOR IMPROVED CAPABILITIES

Extensive interactions with AFSCN MCC and TSC operational personnel during the conduct of Study 232 led to the identification of several categories of possible enhancements to the capabilities of the existing system. Explicit needs have been documented and appear in Reference 2. These capabilities would allow the operators to accomplish their typical tasks more quickly and/or allow operators to be more effective with less training than is currently required. Many of these capabilities started appearing on systems designed in the late 1980s. Specific examples of such capabilities include:

- Ability to use general-purpose utility and analysis software on mission data without having to re-enter data or use totally off-line processing.
- Ability to make changes in selected database parameters quickly and easily, without requiring any software code recompilation.
- Ability to do speed data entry and enhance the automated checking and user prompts to ease use for the operator and minimize data entry errors.
- Ability to provide context-sensitive help, activity support aids, and decision support aids.
- Ability to do historical and trend analysis.
- Ability to customize displays readily and add graphical elements, as appropriate, to support changing operational needs effectively.

MCC ENHANCEMENT OPPORTUNITIES

Overcoming the difficulties, limitations, and challenges identified through discussions with operational personnel is the primary source for MCC enhancement opportunities. Basic automation of time-intensive processes and activities is a crucial first step. This needs to be followed with tools and capabilities that allow operators to accomplish their operations tasks more effectively, with less training, and with personnel that have lower technical skill levels.

On-line documentation, enhanced telemetry server capabilities, improved display and user interface capabilities, activity support aids, and decision support aids are the key elements needed to provide MCC operations improvements. Figure 2 maps these elements to the operations functions and activities previously discussed.

Requirements analysis can be enhanced by providing on-line access to needs documentation and decision support aids that assist in needs prioritization.

Planning can be enhanced by providing on-line access to technical data and procedure information, activity support aids that help in translating needs into corresponding vehicle activity and in building Contact Support Plans (CSPs), and support aids that facilitate mission decisions and selection and ordering of specific vehicle activities.

Scheduling activities can be enhanced by providing displays of schedule information,

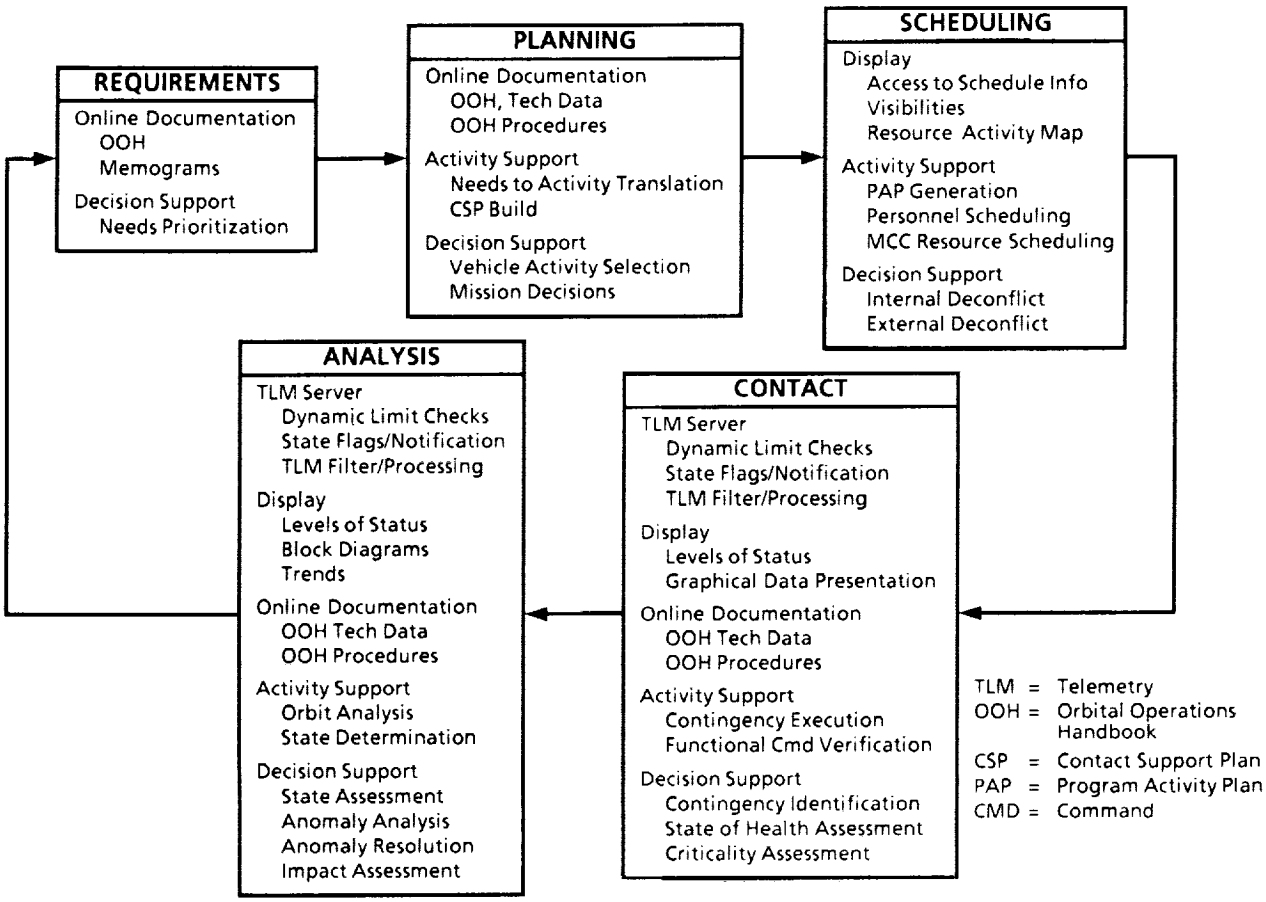


Figure 2. MCC Enhancement Opportunities

satellite/Remote Tracking Station (RTS) visibilities, and resource activity maps; activity support aids that assist in Program Activity Plan (PAP) generation, personnel scheduling, and MCC resource scheduling; and decision support aids that assist with both internal and external resource deconfliction.

Contact activities can be enhanced by providing additional telemetry server functions that provide better limit checking, pattern recognition, and filter/processing capabilities; displays that provide hierarchical levels of status and more graphical presentation of data and procedure information; activity support aids that assist in contingency execution and functional command validation; and decision support aids that assist with contingency identification, state of health assessment, and criticality assessment.

Analysis activities can be enhanced using the same telemetry server, display, and on-line documentation enhancements identified for contact activities, and by providing activity support aids that assist with orbital and attitude analysis and determination; and providing decision support aids that assist with maneuver planning, state assessment, impact assessment, anomaly analysis, and anomaly resolution.

APPLICABLE TECHNOLOGY AREAS

The technology areas applicable to enhancing MCC operations include:

Hypermedia for providing on-line access to space vehicle technical information and procedures.

Telemetry Servers for providing improved telemetry processing and monitoring capabilities.

Advanced Workstations for providing environments for running enhanced support applications.

Databases and Database Management for providing flexible storage and retrieval of a variety of information types.

Expert Systems for providing decision support capabilities.

Human-Computer Interfaces for providing improved display and interface capabilities.

Support Functions for providing basic capabilities such as word processing, spread-

sheets, briefing packages, personal information management, and graphics; as well as more sophisticated analysis tools and training aids such as simulators.

OPERATIONS CONCEPT ISSUES AND ENHANCEMENT DRIVERS

Specific operations tasks vary widely from MCC to MCC, and even from one program to another within the same MCC. The criticality of the need for a particular enhancement is highly dependent on the specific nature, complexity, and difficulty of the planning, monitoring, assessment, and analysis activities supported by the enhancement.

Information overload is rapidly becoming unmanageable. Increasing complexity of space vehicles and increased raw telemetry rates are forcing MCCs to deal with more complex problems in smarter ways. Operators need aids that present information to them in a context that conveys meaning quickly and effectively. Raw telemetry must be converted into meaningful vehicle information that supports the analysis and decision processes.

ARCHITECTURE CONSTRAINTS

Current MCC Architecture

The current MCC architecture was designed in the late 1970s. Its chief features include a closed architecture with alphanumeric workstations tied to planning and evaluation and contact support mainframes. Processing is heavily centralized and display capabilities are limited (see Figure 3).

Transitional Architecture

There are efforts underway to open the architecture (see Figure 4). Improved telemetry and command "front end" processing will free the main processors from some Central Processing Unit (CPU)-intensive functions. The addition of industry standard, high-speed Local Area Networks (LANs) further opens the architecture to decentralization of processing. New, high performance workstations enable critical operations enhancements and superior display capabilities and user interfaces. This also frees the main processors from time-intensive display processing.

Opening the architecture and distributing some of the processing now performed by the main processors must not be done in an ad hoc nor haphazard way. What is required is the evolutionary development of an *integrated space operations support environment.*

INTEGRATED SPACE OPERATIONS SUPPORT ENVIRONMENT (ISOSE)

The major components of the ISOSE are the Environment Manager, Support Functions, Enhanced Display Capabilities, Hypermedia and On-line Documentation Capabilities, Enhanced Database Capabilities, Enhanced Telemetry Front-ends, and CCS Interfaces (reference Figure 5).

Environment Manager

The Environment Manager should provide an integrated environment for virtually seamless control and access to all applications that are a part of the ISOSE. The Human Machine Interface should be primarily graphical with heavy use of windows, menus, and icons--in an environment that supports an object-oriented, user-tailorable, multiple window desktop metaphor. The environment should allow the user to run multiple applications concurrently and provide an easy means for communication between applications.

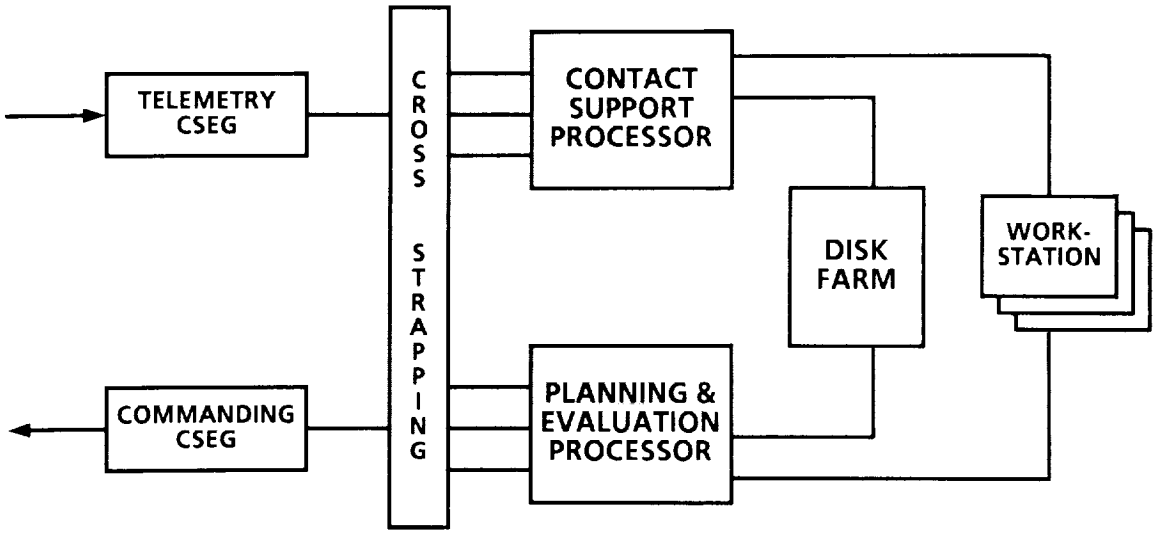


Figure 3. Baseline CCS System Architecture

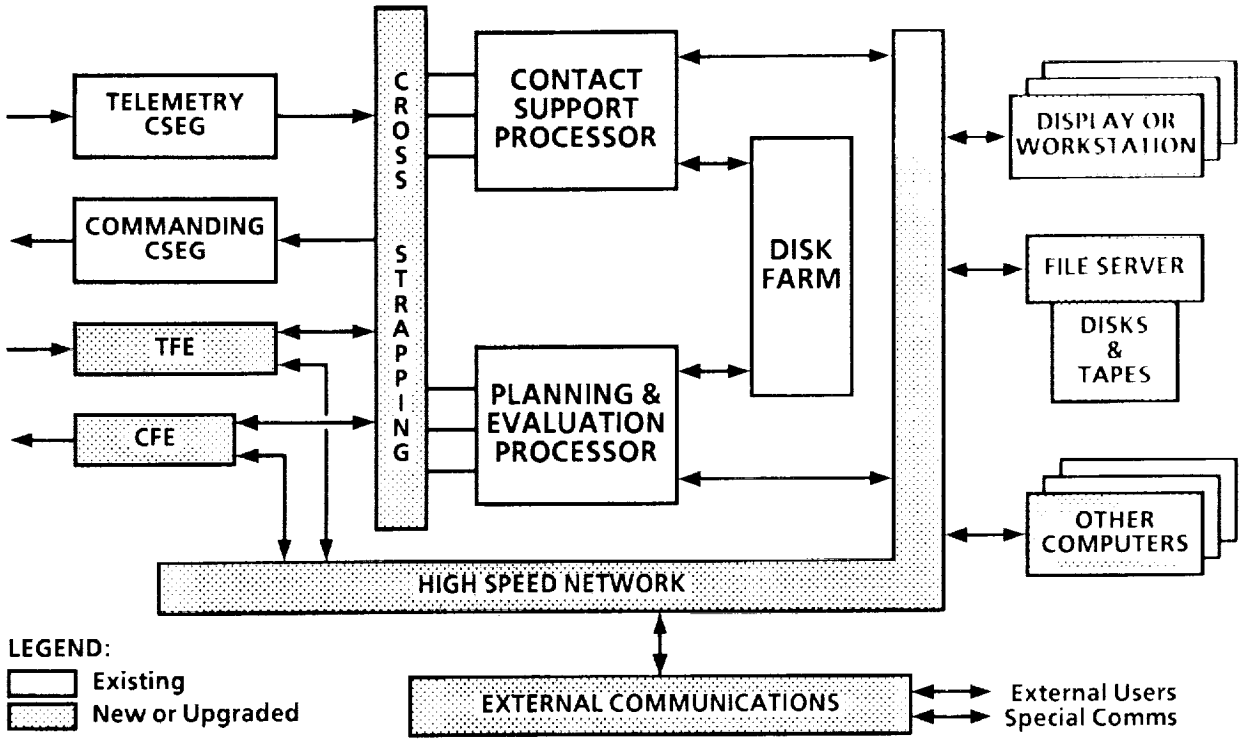


Figure 4. Transitional CCS System Architecture (5 Year Goal)

Support Functions

The major categories of support functions are: 1) Help, 2) General Aids, 3) Readiness Tools, 4) Analysis Tools, 5) Activity Support Aids, 6) and Decision Support Aids.

1. **Help:** Context-sensitive help should be available at all times. This help should be activated and presented in a manner that is consistent throughout all functions and activities supported by the ISOSE.

2. **General Aids:** General aids include an environment/applications/file manager, word processor, spreadsheet, briefing package, general information manager, form generator, and an object-oriented database access capability. These aids should work together in an integrated manner that supports the performance of operations tasks. Additional off-the-shelf software is anticipated, as the capabilities delivered by commercial vendors increase.

3. **Readiness Tools:** These include various training, development, and test tools such as simulators, probes, and database development aids that can assist operators in performing readiness activities.

4. **Analysis Aids:** These include tools that support trending, mathematical analysis, and engineering modeling. These should be easy to use and tailor. Technical analysts should be able to enter their own models without having to write computer code.

5. **Activity Support Aids:**

   a. **Activity Translation Support:** aid to assist operators in translating mission and vehicle needs into the specific vehicle activities necessary to meet those needs.

   b. **Contact Support Plan (CSP) Generation:** aid to automate the generation of specific CSPs that implement desired vehicle activities.

   c. **Program Activity Plan (PAP) Generation:** aid to automate the process involved in generating PAPs. Currently, extensive manual activity is required to transform data through a variety of MCC-specific forms to produce the inputs required for the PAP. This could be greatly simplified by having the forms generated automatically from user inputs, and by having data transcribed from one form to another automatically by the system.

   d. **Personnel Scheduling:** tool to support scheduling of MCC personnel. This task is currently done manually. MCCs typically operate around-the-clock, 365 days per year. This requires scheduling four to five crews under a variety of constraints that vary significantly depending on the nature of operations activities to be supported.

   e. **MCC Resource Scheduling:** tool to support scheduling of MCC resources. Most MCCs have a variety of equipment that can be used

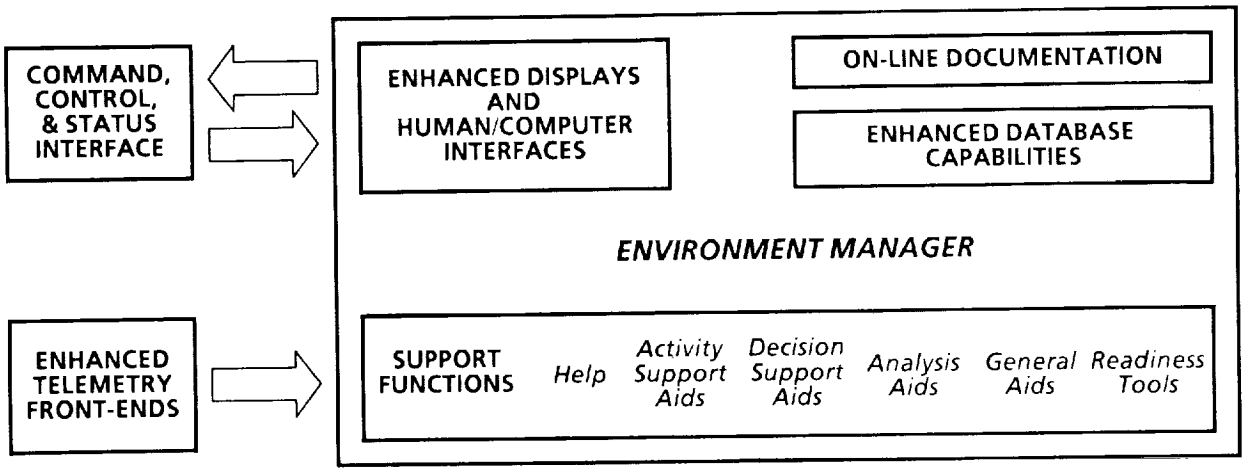


Figure 5. Integrated Space Operations Support Environment

to support multiple programs or crews. In addition, there may be complex dependencies between various modes of simultaneous use and the loading or performance of a particular resource. These common internal MCC resources need to be scheduled and allocated appropriately to the missions and contacts that need them. This task is currently performed manually.

f. **Contingency Plan Execution:** aid to assist and guide operators in the execution of contingency plans, especially for complex contingency plans. The aid should step the operator through the plan, informing the operator of the specific actions he is expected to perform and providing appropriate guidance and recommendations.

g. **Functional Command Verification:** aid to assist operators in monitoring telemetry and identifying functional command verification. This aid should operate synchronously with the CSP being executed.

h. **Orbit Analysis:** specific orbit/attitude analysis improvement needs are being addressed in detail by projects currently underway. Once these projects are completed, an assessment should be made as to whether any additional support is needed for orbit analysis activities.

i. **Status Determination:** aid to assist operators in quickly determining the status and configuration of the ground system and of space vehicle subsystems. Various levels of abstraction should be supported in a manner that allows the operator to navigate easily through the levels to get to the area of interest at the moment. Notifications should be provided to inform operators of specified conditions.

6. **Decision Support Aids:** Decision support aids may be implemented in a variety of ways. In most cases, it is enough to provide an aid that guides the operator through the decision process in a way that facilitates making correct operations decisions. In some cases, the decisions may be straightforward enough that the system can recommend the specific solution and provide a rationale for that choice. The following types of decision support aids are needed:

a. **Needs Prioritization:** aid to assist operators in prioritizing the various mission, health and status, and vehicle needs imposed on them.

b. **Vehicle Activity Selection:** aid to assist operators in selecting particular vehicle activities that can meet various needs, objectives, and operations constraints.

c. **Mission Optimization:** aid to assist operators in optimizing mission performance given a current or expected condition, a set of applicable constraints, and a time limit for coming up with an acceptable solution.

d. **Internal Schedule Deconfliction:** aid to assist operators in removing conflicts from the internal MCC schedule. These conflicts typically require shifting planned activities, rescheduling, and reallocating resources. The deconfliction aid should assist operators in finding acceptable solutions that can be achieved without drastic changes to current planned activities.

e. **External Schedule Deconfliction:** aid to assist operators in removing conflicts from the external AFSCN schedule. These conflicts typically require shifting planned activities, rescheduling, and reallocating resources. The deconfliction aid should assist operators in finding acceptable solutions that can be achieved without drastic changes to current planned activities.

f. **Contingency Identification:** aid to assist operators in categorizing the current operations situation and identifying the

appropriate contingency that applies. Typically, the operator needs to deduce something about the nature of the problem before a contingency procedure can be selected. This deduction is not always obvious from the telemetry indications.

g. **State of Health Assessment:** aid to assist operators in assessing the state of health of the systems for which they are responsible. This aid should automatically monitor appropriate telemetry parameters and notify the operator of any anomalous indications and what they may mean. This aid should permit the creation of hierarchies of notification that assist the operator in identifying and paying attention to what is most critical at the current time.

h. **Criticality Assessment:** aid to assist operators in assessing the criticality of an anomalous situation. It should help operators to decide what operations options are most appropriate (fix, diagnose, collect data, mitigate impacts, wait, call experts).

i. **Impact Assessment:** aid to assist operators in assessing the impacts of a current situation or planned course of action. It should help to identify what risks are involved as well as what benefits and degradations should be expected.

j. **Anomaly Analysis:** aid to help operators through the analysis process for various anomalies. It is expected that these aids need to be tailored to the diagnosis and analysis processes required for specific kinds or categories of anomalies or for specific kinds of vehicle subsystems.

k. **Anomaly Resolution:** aid to help operators through the anomaly resolution process. These should work in conjunction with aids that support the anomaly analysis process. It is expected that these aids need to be tailored to the resolution processes required for specific kinds or categories of anomalies or for specific kinds of vehicle subsystems.

Enhanced Display Capabilities

Enhanced display capabilities include user tailorable telemetry displays, graphics capabilities, and standard window and menu capabilities. These capabilities should allow the operators to generate custom displays that include text, graphics, and user interaction.

Hypermedia and On-line Documentation Capabilities

Hypermedia provides a means for electronically capturing and accessing text, graphics, video, and sound in a flexible and intuitive manner. It permits very fast and flexible access to the captured data. Some of the operations data that should be accessible electronically includes:

Space Vehicle Technical Data:
- Orbital Operations Handbooks
- Memograms
- Other Technical Data

Interactive Information:
- Procedures
- Contingency Plans
- Generic Contact Support Plans

Enhanced Database Capabilities

Database capabilities provide the means for storing and retrieving a variety of kinds of information. As a minimum, the ISOSE should provide flexible capabilities for operators and applications to generate, modify, and access data from the following:

Telemetry Databases:
- Telemetry format database per program
- Telemetry history database per vehicle
- Telemetry constraints and checks per vehicle

Command Databases:
- Command format database per program
- Command history database per vehicle
- Command sequence constraints and checks per vehicle

Activity Database:
- Contact support plans per contact
- Activity log data per contact
- Special limit checking and notification data
- Special activity and decision support aid definitions

Display Database:
- Operator telemetry display definitions
- Help display definitions
- Activity support display definitions
- Decision support display definitions

Enhanced Telemetry Front-Ends

Enhanced telemetry front-end capabilities should provide dynamic, user-tailorable limit checking, state flags and notifications, telemetry filtering and processing, pattern detection, and dynamic user-selectable monitoring and display of telemetry parameters.

Command, Control, and Status Interface (C/C/S)

The ISOSE will have to use current C/C/S links, or their equivalents, to get control directives out to AFSCN communications and RTS equipment; to get commands out to the RTSs for transmission to the space vehicle; and to observe status information from various AFSCN resources.

CONCLUSIONS

There is an overwhelming need for a variety of MCC/TSC enhancements in the 1990s. Basic automation of time-intensive manual processes is a crucial first step. This needs to be followed with

tools and capabilities that allow operators to accomplish their operations tasks much more effectively.

The technology areas most applicable to developing needed MCC/TSC enhancements include hypermedia, telemetry servers, advanced workstations, database and database management, expert systems, human-computer interfaces, and a variety of support functions.

The Integrated Space Operations Support Environment provides a conceptual architecture that addresses key enhancement requirements. The major components of this enhanced operations environment are an environment manager; enhanced display capabilities; hypermedia and on-line documentation capabilities; enhanced database capabilities; enhanced telemetry front-ends; enhanced C/C/S interfaces; and a variety of support functions integrated into the environment. These support functions include help, general aids, readiness tools, analysis aids, activity support aids, and decision support aids.

Most of the MCC/TSC enhancements identified are well within the state of current technology. The challenge is to apply these technologies to specific space operations improvement needs in a manner that results in low-risk, cost-effective, new capabilities that can be integrated into the AFSCN Command and Control System Architecture.

REFERENCES

1. Bannister, J. A., Frazier, Jr., E. R., and Lee, C. A., The Aerospace Corporation, "Guidelines for the Evolution of the CCS Architecture," Contract F04701-88-C-0089, Aerospace Report No. TOR-0091 (6489-05)-1, Revision 1, El Segundo, California, 4 April 1991.

2. Golden, M., Hartman, W., Monosoff, S., and McDaid, P., Loral Space & Range Systems, "Enhancing Mission Control Center Operations Using Automation and Expert Systems Technology," Contract F04690-86-C-001, CDRL 023A2, Loral Report No. CUE-TR-90-135, Sunnyvale, California, 31 January 1991, (Study 232 Report).

SOAR
Wed, May 22, 1991

The Real Time Data System (RTDS) project of the Mission Operations Directorate (MOD) of NASA's Johnson Space Center has been developing real-time expert systems to support flight critical decision-making in the Mission Control Center for the past four years. These expert systems provide deterministic and heuristic analysis of Space Shuttle telemetry data and provide the results to flight controller operators at consoles in Mission Control. The primary purpose of these expert systems is fault diagnosis, isolation and recovery. The primary goal of these expert systems is to increase the quality of flight decision making.

The initial goal of the RTDS project was to demonstrate the viability of expert systems and advanced automation in a real operations environment. As a technology demonstration project, RTDS has been a clear success for NASA and an indicator that expert systems can and should play an important role in critical operations such as manned space flight. After four years of use as operator assistant tools, these applications are now being transferred to operational status and are becoming main-line items critical to mission success. With the RTDS project, NASA has demonstrated the ability to transfer leading edge technologies from a laboratory environment and place them on line in a mission critical environment at a reasonable cost.

Session I6: SOFTWARE ENGINEERING

Session Chair: Dr. Kirstie Bellman

N93-11943

THE DEVELOPMENT AND TECHNOLOGY TRANSFER OF SOFTWARE ENGINEERING TECHNOLOGY AT JOHNSON SPACE CENTER

C. L. Pitman, D. M. Erb*, M. E. Izygon**, E. M. Fridge III, G. B. Roush, D. M. Braley, and R. T. Savely
NASA/Johnson Space Center
Software Technology Branch/PT4
Houston, TX 77058

ABSTRACT

The Unites States' big space projects of the next decades, such as Space Station and the Human Exploration Initiative, will need the development of many millions of lines of mission critical software. The Johnson Space Center (JSC) is identifying and developing some of the Computer-Aided Software Engineering (CASE) technology that NASA will need to build these future software systems. The goal of this research and development is to improve the quality and the productivity of large software development projects. This paper outlines new trends in CASE technology and describes how the Software Technology Branch (STB) at JSC is endeavoring to provide some of these CASE solutions for NASA. Key software technology components include knowledge-based systems, software reusability, user interface technology, reengineering environments, management systems for the software development process, software cost models, repository technology, and open, integrated CASE environment frameworks. The paper presents the status and long-term expectations for CASE products. The STB's Reengineering Application Project (REAP), Advanced Software Development Workstation (ASDW) project, and software development cost model (COSTMODL) project are then discussed. Some of the general difficulties of technology transfer are introduced, and a process developed by STB for CASE technology insertion is described. Finally, plans for future work are outlined.

1. INTRODUCTION

The Space Station Freedom Program and the Human Exploration Initiative will require the development, use, and maintenance of software systems containing millions of lines of code. These software systems present severe technical and managerial challenges. For example, the development and maintenance of these systems will require hundreds of well-trained software engineers, often working in parallel, and this will require well-managed, disciplined software development and maintenance processes. NASA is aware of these challenges and understands that new software engineering technology will be needed in the coming years. This paper describes ongoing work within the Software Technology Branch (STB), Johnson Space Center (JSC), to identify and develop some of the software technology required to meet these challenges and to transfer it to all of NASA.

The main goals of this work are to improve the productivity of software developers, users, and maintainers and to improve the quality of the software systems. This is to be accomplished through the enforcement of good software engineering principles and the use of Computer-Aided Software Engineering (CASE) throughout the whole software life cycle. Within an open, fully-integrated CASE environment, knowledge-based technology will be used to guide the entire software development process. Where necessary, large existing programs that have become too costly to maintain will be reengineered. Software and data reusability and application generators are also key elements of this approach to improve software quality and productivity.

2. SOFTWARE ENGINEERING TECHNOLOGY

One of the biggest thrusts in software engineering technology today is the push to develop an open, fully-integrated CASE environment -- i.e., a CASE environment wherein different vendors' hardware and software components can work together effectively. Many of the world's largest computer companies (such as IBM and DEC) and software vendors, the U.S. Department of Defense (DOD), and others are investing great sums in research and development to produce a *framework* to support an open, fully-integrated CASE environment. A reference model for CASE environment frameworks has been developed by the European Computer Manufacturers' Association (ECMA), and it has been submitted as a proposed standard to the National Institute of Standards and Technology (NIST). This model (figure 1) will be referred to as the NIST/ECMA Reference Model [1], but it is often informally called the "toaster model" because of its general appearance.

For the purpose of this paper, the NIST/ECMA reference model will be used to serve as an introduction to software engineering technology and as an illustration of how the different projects within the Software Technology Branch (STB) fit into the "big picture" of CASE environments (see figure 1). Therefore, it is useful to give a short description of the NIST/ECMA reference model for a CASE environment framework, but first some important software engineering concepts and some introductory background and definitions are presented.

2.1 Definition of Software Engineering

Software engineering may be defined as the planned process of producing well-structured, reliable, good-quality, maintainable software systems within reasonable budgets and time frames [2]. A few aspects of this definition need elaboration.

1) Software engineering is engineering. In particular, software engineering involves the application of systems engineering principles and techniques to the development of software systems. A methodical systems engineering approach is especially important in the case of large software systems which are typically extremely complex and require large teams of professionals to develop and maintain.
2) Software engineering involves both *product* and *process* [3]. A well-engineered software product is *documented* software that provides the services required by its users and which is maintainable, reliable, efficient, and provides an appropriate user interface. The software engineering process involves both technical and non-technical issues. As well as knowledge of specification, design and implementation techniques, software engineers must have some knowledge of human factors and software management [4].
3) Software engineering strives to produce cost-effective software systems via a cost-effective process. The software should be developed and maintained using appropriate cost, time, and personnel resources.

---

* The MITRE Corporation
** National Research Council Research Associate

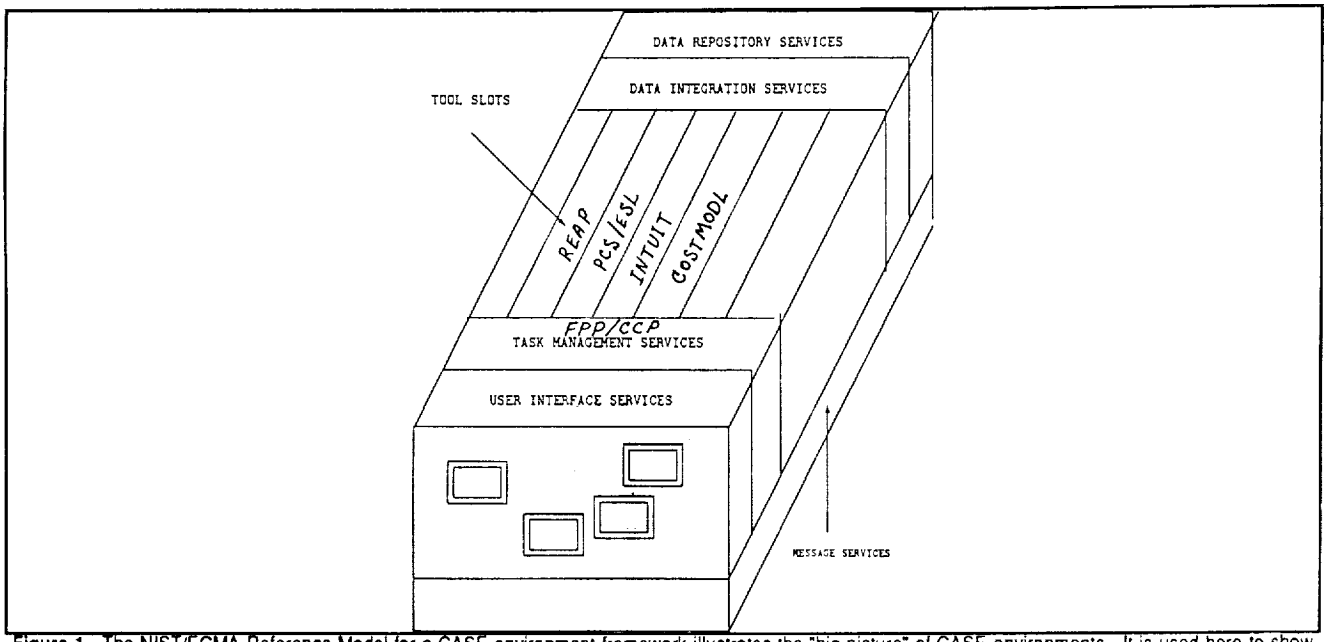


Figure 1. The NIST/ECMA Reference Model for a CASE environment framework illustrates the "big picture" of CASE environments. It is used here to show where the different projects of the Software Technology Branch (STB) fit in that big picture. (Note that the reference model does not recommend specific tools, and the STB tools shown here are not a part of the reference model [1].)

Learning how to be a good software engineer only begins with learning how to generate computer programs [5].

2.2 The Software Engineering Process

The phases through which a software system evolves during its lifetime, from the earliest exploratory phase in which the feasibility of the proposed system is explored to the phaseout stage in which the software system is discontinued, is referred to as the **software life cycle**. NASA's Software Management and Assurance Program (SMAP) has defined a standard, tailorable, Software Acquisition Life Cycle [6] that is composed of eight phases: concept and initiation, requirements, architectural (preliminary) design, detailed design, implementation, integration and testing, acceptance and delivery, and sustaining engineering (commonly called "maintenance") and operations.

Reports vary on the actual figures, but industry spends from 40% to 75% of its total hardware and software budget in the software maintenance phase alone [7]. This is because of changing requirements that arise from a changing environment and growing expectations. If software systems could be engineered (or reengineered) so that they are easier to understand and maintain, substantial savings might be realized during the maintenance phase. To help accomplish this goal, a specific software development **method** (i.e., a formal set of procedures or guidelines) should be employed during each life-cycle phase to produce the products of that phase. For example, there are three classes of methods applicable to the architectural and detailed design phases: top-down structured design, data-structure design, and object-oriented design [8]. Also, there is a need to capture metrics to evaluate how well a software development organization is performing, to track the quality of the software produced by the organization, and to determine whether changes result in actual improvements to the product and process.

The Software Engineering Institute (SEI) at Carnegie-Mellon University has developed a "Software Process Maturity" model (figure 2) for describing the maturity levels of software development organizations [9]. Most organizations have a very low maturity level (1), and only very few have a high maturity level (4 or 5). If the key problem areas at a given level are resolved by an organization, then that organization has progressed to the next higher maturity level in the model. Note the types of problem areas that typically exist in organizations at the various levels. In particular, automation (although potentially very valuable) cannot *by itself* improve an organization's software process maturity (at least, not until level 5 is attained).

2.3 Introduction to CASE

Computer-Aided Software Engineering (**CASE**) tools *assist* the software engineer in the application of software engineering methods. In the same sense that a word processor supports an author, a CASE tool provides technological support to a software engineer for drawing complex diagrams, for checking syntax and semantics, and in general for implementing a specific method efficiently. Clearly, a CASE tool cannot replace the software engineer anymore than a word processor can replace an author. Furthermore, CASE tools cannot assist in the software engineering process if such a process does not exist within a software development organization (i.e., if the organization's maturity level is 1). In such a situation, the organization must first adopt a software engineering process before it buys CASE technology; otherwise, the purchased CASE tools will rapidly become "shelfware."

2.4 Components of a CASE Environment

A **CASE environment** is an information system that provides support for software engineering. A CASE environment deals with information about software under development (such as project plans, requirements, designs, specifications, source code, and test data) and about an organization's software engineering process. Some alternative names for a CASE environment are Integrated Project Support Environment (IPSE) and Software Engineering Environment (SEE).

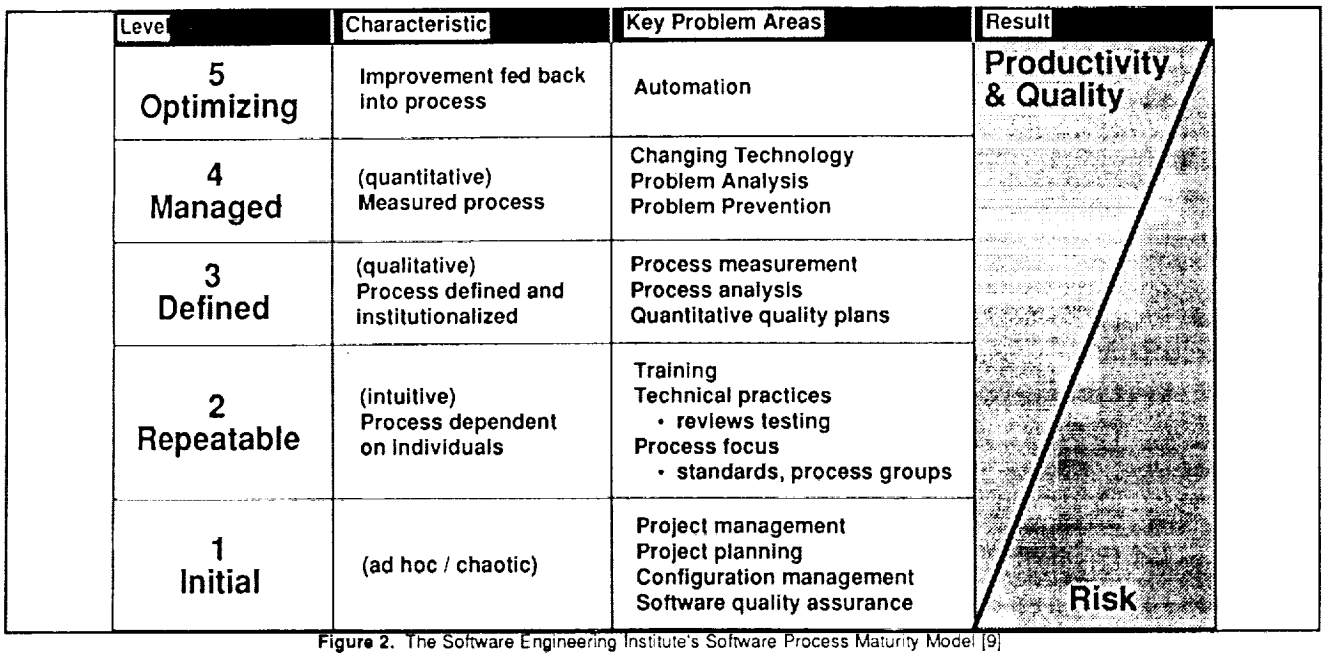


Figure 2. The Software Engineering Institute's Software Process Maturity Model [9]

A CASE environment consists of a relatively fixed set of core facilities, called the **environment framework**, and a set of facilities called **tools**, which are specialized for particular CASE environments and which are not available in all environments [1]. Tools that are **fully integrated** into the environment framework use the services provided by the environment framework extensively and may also use services provided by other tools. In contrast, **loosely integrated** tools use very few (if any) of the framework's services and do not use services provided by other tools.

2.5 The NIST/ECMA Reference Model

The tool slots shown in figure 1 represent the concept of sets of tools that can be readily integrated into a CASE environment framework in order to create special-purpose environments. The NIST/ECMA reference model does not advocate particular tools nor does it classify tools. However, two general categories of tools are classically defined: **vertical tools** (that are applicable during a single phase of the software development life cycle) and **horizontal tools** (that are applicable across the entire software life cycle). For example, a compiler and a code generator are vertical tools, but a project management tool is a horizontal tool.

Figure 1 is NOT meant to represent a functionally-layered model wherein one layer can only interface with adjoining layers. The reference model is simply based on grouping sets of framework services together in such a way that each grouping may be expected to be covered by existing or future standards. This aids in the definition and evolution of standards, a major goal of the NIST/ECMA reference model. This grouping of services also enables various kinds of integration to be discussed: **presentation integration** (user interface services), **control integration** (task management services plus the message services), and **data integration** (data repository services plus data integration services). The Appendix gives a brief description of each of the major groups of services.

Some of the aims of the NIST/ECMA reference model are:

1) provide a basis for describing, comparing, and contrasting existing and proposed environment frameworks;
2) provide a means for the smooth and coordinated evolution of future standards for environment frameworks;
3) address interoperability and integration of tools;
4) cover all framework aspects irrespective of implementation techniques or software development methods.

2.6 The Software Technology Branch's CASE Projects

As mentioned earlier, the NIST/ECMA reference model will be used in this paper to serve as an introduction to software engineering technology and as an illustration of how the different projects within the Software Technology Branch (STB) fit into the "big picture" of CASE environments (see figure 1). The rest of this paper discusses each of the STB's CASE projects.

- The STB is endeavoring to track the major CASE environment research and development efforts around the world, in order to get a better understanding of the big picture.
  - Section 3 gives the results of an analysis of today's CASE products and estimates the long-term prospects for CASE products.

- Other STB projects deal with tools that "plug into" the tool slots in figure 1.
  - The STB's Reengineering Application Project (REAP) is discussed in section 4.
  - The Advanced Software Development Workstation (ASDW) project is discussed in section 5, including descriptions of the reuse-oriented Parts Composition System (PCS) and Engineering Script Language (ESL) in 5.1, and the INTelligent User Interface development Tool (INTUIT) in 5.2.
  - The software development cost model (COSTMODL) project is discussed in section 6.

- The Framework Programmable Platform (FPP), a subtask of the ASDW project, is researching some of the environment framework's Task Management Services (see figure 1).

- The FPP subtask is developing a horizontal tool, called a Configurable Control Panel (CCP), for specifying, managing, and enforcing a model of the software development process. The FPP/CCP is discussed in section 5.3.

- Technology transfer of emerging CASE technology to NASA is the primary objective of all these STB projects.
  - The technology transfer process is discussed in section 7.

3. COMPUTER-AIDED SOFTWARE ENGINEERING: TODAY AND TOMORROW

As discussed in section 2, many major research and development programs are attempting to develop an open, fully-integrated CASE environment. Of course, the Software Technology Branch (STB) does not have the resources to develop such an environment on its own, nor is this necessarily desirable. However, the STB is attempting to identify those areas where CASE technology is approaching sufficient maturity that it can be transferred to NASA. Additionally, the STB is identifying those areas where CASE research and development show significant promise. In a few of these promising areas, NASA can leverage ongoing research and development work and thus ensure the development of the software technology that it will need to meet its own particular challenges.

A major project within the STB is the analysis of CASE technology today and the estimation of long-term prospects for CASE products. In a few cases, this analysis has even included very detailed, hands-on evaluation of prototype CASE environment frameworks, such as the United States Air Force's Software Life Cycle Support Environment [10,11].

3.1 Current Status of CASE Products

Today, there are hundreds of CASE vendors marketing tools. These vendors have heard the customers' demand for an integrated set of tools supporting software development activities across the full life cycle. The response has been in a variety of forms:

1) a vendor defines its set of tools as "full life cycle" (caveat emptor!);
2) a vendor buys or licenses another vendor's tools to create a more complete set of tools for the life cycle;
3) a vendor signs up as a business partner or program participant in one or more of the major corporations' CASE environment projects; or
4) a vendor becomes active in the standards meetings that are defining the various interfaces involved in a truly integrated CASE environment.

There are several messages to be understood in these different responses. The market consolidation indicates that no vendor, not even a major corporation, is willing to assume the total risk of investing in a completely proprietary CASE environment. Another message is that the educated consumer recognizes the dynamic nature of the technologies supporting CASE evolution. The STB is in the role of ensuring that JSC consumers do not buy CASE products which have no potential for evolution (upgrade).

Today's CASE consumer can buy the following:

1) tools that function on stand-alone, networked, mainframe-cooperative, or mainframe-dependent PCs and workstations;
2) tools that allow users on networked platforms to work with them simultaneously;
3) integrated tools from a single vendor that support the development of Management Information Systems (MIS) reasonably completely across the phases of the life cycle; and
4) tools for the development of Aerospace/Defense Engineering (ADE) systems. [These ADE tools are more limited and less integrated than their MIS counterparts. Some of these tools support DOD standards (e.g., MIL-STD 2167A). None support NASA standards (e.g., SMAP).]

The reasons for this situation are economic. 80-90% of the software developed today is business-oriented rather than engineering-oriented. Also the DOD has been funding software engineering methods and their implementation for 15-20 years. Nonetheless, NASA can leverage this work to its benefit.

For MIS applications, CASE technology can provide many reasonable solutions today. However, an open, fully-integrated CASE environment for the development of MIS does not yet exist, mainly because repository technology (for storing the megadata produced by CASE environments) is not yet mature enough.

There are some major deficiencies today in commercial CASE support for ADE systems. These deficiencies include:

1) quality of implementation of the method(s) to be used in a phase of the life cycle;
2) general lack of automated support for the generation of test scripts;
3) little or no assistance for reverse engineering of assembly code;
4) lack of adequate project management capabilities, including the lack of metrics which support project management; and
5) lack of a totally satisfactory notation for unambiguously expressing the design of a distributed system with multiple, asynchronous events.

There are also many positive signs in CASE products today for ADE and especially for real-time systems. The most significant of these is the capability to simulate the behavior of the system during the design stage from Program Design Language (PDL) generated on the basis of diagrammed behavior. Another significant capability of some CASE products is the ability to address the total system design. Large, complex systems are built to operate with hardware, software, and people. CASE tools for system co-design are becoming available in which components need not be specified as either hardware, software, or people functions in the early levels of the design.

3.2 Long-term Expectation for CASE Products

The unavailability of the special-purpose, truly-integrated CASE environments that are ideal for some applications will exist for at least another five years. Within five years, important interface standards will have been agreed upon and some environment frameworks will be robust enough for collections of compatible tools to be installed into a customized CASE environment. This will be a major step forward. However, integrable, plug-in horizontal tools will come later because they must cross the total life cycle development process in order to provide tailored documentation, project management, and configuration management and to measure the quality of the software engineers' work. The repository technology, especially the repository management system, must mature in order to provide robust support for horizontal tools.

Within ten years, mainframes may still be used for software execution but are unlikely to be used extensively for software development. Reliable, lower-cost, distributed approaches will likely be dominant. Some CASE environments may be globally distributed. Security issues for distributed systems will be

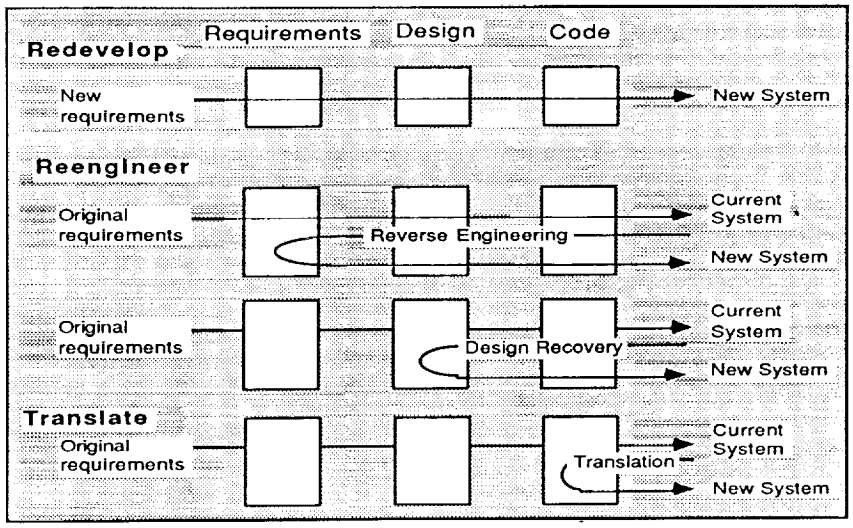


Figure 3. Alternative solutions to updating software systems to modern software engineering standards

resolved. Some CASE environments will have a rule base (or knowledge base) that can be configured with an organization's software development process in order to help manage the process. The project manager will assign responsibilities to his/her staff, constrain their access accordingly through rules, and automatically measure their skill and productivity against a variety of historical statistics. Reuse libraries will be well-categorized, accessible from a CASE environment, and the concept of synthesizing systems from "look-a-likes" will be supported. Many repetitive tasks will be automated. Much manual work will still be needed, but it will be concentrated in the systems engineering, specification, and design activities.

4. SOFTWARE REENGINEERING

The second important research project underway at the STB is the Reengineering Application Project (REAP). An environment (or, more accurately, an integrated tool set) for the reengineering of Fortran programs has been identified by the STB as a promising CASE area which, although requiring additional development, can leverage resources already developed for NASA.

Many Fortran systems developed in the 1960s and 1970s represent an enormous investment for the JSC. Today, these software systems are as crucial for the space program as they are expensive to use and maintain. This problem has led the STB to look for possible solutions and to consider the evolution path of these systems during the next 5 to 10 years. Among the three primary alternative solutions (i.e., complete redevelopment of the programs, code translation to a more modern language, and reengineering), reengineering appears to be the most promising path (figure 3). The goal of reengineering is to update a program to modern software engineering standards without losing required function and data and without losing the engineering knowledge embedded in the code. An integrated tool set for reengineering Fortran systems is needed to accomplish this goal efficiently. Currently, only a few commercial tools exist that aid in the reengineering of Fortran programs The STB is not only investigating these tools but is also researching and developing the capabilities needed to completely support the reengineering of Fortran systems.

4.1 Definitions

Reengineering is the combination of the reverse engineering of a working software system followed by the forward engineering of a new system based on the results of the reverse engineering.

Forward engineering is the standard process of developing software from requirements. It is composed of many life cycle phases such as requirements, architectural design, detailed design, coding, and testing.

Reverse engineering is the reverse of forward engineering. It is the process of starting with existing code and going backward through the software development life cycle. Life cycle products are obtained by abstracting out only the essential information and hiding the non-essential details at each reverse step.

How far to go backward in the reverse engineering process before it is stopped and forward engineering begins is a critical question and involves trade offs. It is important to understand all of what the program does, all of the information it handles, and the control flow. In other words, the reverse engineering process must be carried far enough back to understand *what* the "as is" program is.

Reverse engineering is referred to as **design recovery** when the reverse engineering process stops at the recovery of the design rather than proceeding on to a higher level of abstraction to include the recovery of the requirements. The basic process of design recovery involves recovery of information about the code modules and the data structures in an existing program. This information can support the programmer/analyst(s) who is either maintaining a large, unfamiliar Fortran program, upgrading it for maintainability, or converting it to another target language.

Of course, a better job of redesigning a program can be accomplished when the reverse engineering process is carried beyond design recovery to **requirements recovery**. However,

requirements recovery is difficult and requires higher levels of domain knowledge to do the abstractions. The *whys* of the requirements, design, and implementation can only be provided by someone very familiar with the program and the domain. This level of expertise is often very difficult to find and to have dedicated to the reengineering process.

4.2 Reengineering Strategy

The STB is proposing standards, methods, and an integrated reengineering tool set [12,13] based upon the significant set of tools built to develop and maintain Fortran programs for the Space Shuttle [14,15]. The proposed tool set must support these standards and methods even in areas where the language definition and compilers do not enforce good software engineering practices. The intent is to get an integrated tool set out into use in JSC's maintenance community to provide support for upgrading Fortran programs in terms of maintainability in the near term, and then to extend the functionality of the tool set in response to feedback from the programmers/analysts. Later versions of the integrated tool set may have extensions to handle programs written in C, Ada, or even HAL/S, according to requests from the user community.

The STB has defined new standards for its Fortran programs [12]. These new standards are added to previously defined standards for Fortran programs which specified coding standards, in-line documentation standards, global data structure standards, and unique data naming conventions. The new standards produce a Fortran program with good software engineering structures such as those found in Ada. These new Fortran standards address documentation, longer variable names, modern control flow structures, grouping of subroutines into virtual packages, data structuring, and separation of input/output processing from the principal functionality provided by the program.

The reengineering methods developed by the STB are aimed at progressively converting a Fortran program into more maintainable states [12]. They define the way to convert an arbitrary Fortran program to the STB's previously defined Fortran standards, then to the new standards, and finally to a target language that embeds software engineering principles (such as Ada). These reengineering methods define the steps and the skills required and give guidelines on how far to reverse engineer before deciding to rebuild.

The STB's proposed integrated reengineering tool set [12] will support the above standards and methods. A preliminary tool set has been developed that supports the maintenance of Fortran programs and that aids in the initial analysis phase of reengineering (to Fortran 90, C, or Ada). It contains modified versions of the tools used to support the current STB Fortran standards, plus commercial off-the-shelf (COTS) tools, and additional custom-built tools. The tools are integrated at the front end by a user interface (i.e., presentation integration) and behind the screen by two logical databases, an inter-tool database and a source code database. However, COTS tools cannot be integrated seamlessly into the tool set at this time. This tool set will not completely automate the reengineering process since much reengineering work must still be done by a programmer/analyst. However, as an experience base is accrued in design recovery, knowledge-based capabilities can be added to assist the programmer/analyst even further.

5. THE ADVANCED SOFTWARE DEVELOPMENT WORKSTATION PROJECT

As discussed in section 2, the NIST/ECMA model of a CASE environment framework does not recommend specific tools nor does it classify them. The STB's analysis of CASE product status and direction indicates that a number of good tools are available, but there are some important capabilities and tools that still require further research and development. The Advanced Software Development Workstation (ASDW) project is researching and developing specific types of advanced technology and tools that an advanced workstation for software development should provide [16]. One of these tools is a Parts Composition System (PCS) for developing applications from reusable software parts, using knowledge-based technology. Another tool being developed is the INTelligent User Interface development Tool (INTUIT). A third tool, a Configurable Control Panel (CCP) for an integrated CASE environment, is being developed under the Framework Programmable Platform (FPP) subtask of the ASDW project. The CCP will be a horizontal tool for managing and enforcing a (locally configurable) model of the software development process.

5.1 Parts Composition System and Engineering Script Language

Many researchers think that software reuse is a way to significantly improve the quality of applications and the productivity of application developers. The ASDW's Parts Composition System (PCS) is a set of tools for developing applications from reusable software parts. Before describing the PCS, however, one aspect of the concept of compositional reusability demands a brief explanation.

A library of procedures (or, more generically, primitives) containing the reusable software parts must be available before they can be put together to form a complete application; i.e., the components must exist before they can be assembled. It is very important to understand that not just any procedure can qualify as a *reusable* software part. Reusable primitives must be specifically designed for reuse. They must be optimally designed to strike a balance between the desire for general applicability and the need for applicability to a given class of problems. In other words, reusable parts must be carefully engineered so that they can be used throughout an explicitly specified domain.

The development, organization, and maintenance of these domain-specific libraries of reusable parts is primarily the responsibility of the software development engineers and not the job of the application developers, who may be aerospace engineers with minimal programming experience. The software development engineers receive part specifications from the application developers and provide implementations to populate the required libraries. If well managed, this separation of roles helps to limit the amount of domain expertise that the software developer must have and also the amount of programming experience that the application developer must have.

Once an application developer has selected the most appropriate domain-specific library of parts, he/she invokes a PCS tool called the Engineering Script Language (ESL) editor. This is a graphical editor that allows the application developer to create, modify, store, and retrieve graphs that represent applications. The graphs show the structure of an application (i.e., how it is made from its component parts) and what data and controls flow between the components. The components are depicted by boxes called nodes, and the data and controls are shown as arrows linking the nodes. Other structures allow for merging and replicating links and for including looping and branching logic. Each component (box) is either a primitive from the library or a subgraph, which makes possible hierarchical decomposition.

Once the graphs representing an application are completed, the application developer will invoke menu commands to validate the graph system and to generate the required code in some high-order language, such as Ada. The generated code, in the form of a main program and subprograms, will then be ready to be compiled and linked with the object code of the primitives from the

domain-specific library. Alternatively, source code templates (such as Ada generics or even main programs with certain parameters that must be initialized before compilation) might be generated, if required.

The internal representation and storage of the ESL graphs, the semantic interpretation and validation of the graphs, and the generation of code in a high-order language will be done using knowledge-based technology. To date, specifications have been developed for the ESL system. Implementation of the first ESL prototype is underway.

5.2 INTelligent User Interface Tool

A good user interface is critical to the successful use of a complex scientific application such as a space flight simulation, which typically involves very large sets of input data. Even an expert user may expend substantial effort to introduce the right data in the right manner. An Intelligent User Interface (IUI) uses knowledge-based technology to provide the user with the capability to easily prepare the input data without requiring prior extensive knowledge of the underlying software. An IUI is also commonly called a Knowledge-Based Front-End (KBFE). INTUIT (INTelligent User Interface development Tool) is a generic IUI shell that a knowledge engineer configures for a specific application by adding a knowledge base that includes input variable names which are immediately understandable by the users, the range of permissible data values, the structure and format of the data sets, and rules for error and consistency checking. The current knowledge representation scheme used within an INTUIT knowledge base is fully described in [17]. Many of the same subsystems required by a PCS are also required by INTUIT, which may therefore be considered to be a "PCS for input data sets." In fact, INTUIT is a PCS subshell.

INTUIT's user-friendly interface relies on the Transportable Applications Environment Plus (TAE+) [18,19] developed by NASA/Goddard Space Flight Center to provide a graphical windowing environment with a mouse and icons. TAE+ is a very powerful graphical interface development tool, built as a layer on top of X-Windows (from M.I.T.). The graphical interfaces built using TAE+ are very portable. INTUIT provides standard panels for the user to browse and modify objects in the knowledge base. In addition, INTUIT allows the knowledge engineer to design "custom forms" or input screens using the TAE+ Workbench and to describe these forms via schemas in the knowledge base. This capability gives the knowledge engineer a way to specify good data organization and to enforce it, as well as providing a modern user-friendly interface with point and click capabilities.

The INTUIT shell was used to develop a KBFE for Space Vehicle Dynamics Simulation (SVDS), a computer program currently used at JSC for designing the trajectory and flight plans for Space Shuttle missions. The SVDS application called Ground Simulation (GNDSIM) [20] was selected for KBFE development, and an INTUIT knowledge base was built for it. Flight planners use GNDSIM to verify and refine the sequence of maneuvers required to accomplish a rendezvous. A thorough discussion of the development and testing of this KBFE for GNDSIM is presented in [21], but the results can be summarized as follows. All the users who participated in the tests were very satisfied with the KBFE. Building an input data stream with the KBFE proved to require from one-half to one-fifth the time needed using the current interface. As a result of these tests, specific enhancements to INTUIT are planned. The development of KBFEs for other tools used by the flight designers is also being considered.

5.3 Framework Programmable Platform

The Framework Programmable Platform (FPP) subtask of the ASDW project is researching some of the issues involved in building an integrated CASE environment. The current focus of the FPP is the management and control of the software development and maintenance processes -- crucial factors in the success or failure of any large software system. In the NIST/ECMA model (figure 1), the services that the FPP is focusing on are part of the Task Management Services. Specifically, the FPP subtask is developing a horizontal tool, called a Configurable Control Panel (CCP), for specifying, managing, and enforcing a model of the software development process.

The CCP, which captures an organization's experience and best-practice rules of software development, will be configured by a system administrator/manager of the software development organization and will define for each step of the software development process [22]:

1) the minimum tasks that must be performed to complete the step,
2) the methods, tools, and computer resources available to projects at the local site,
3) the personnel roles that can be assigned to project personnel, and
4) the configuration control method(s) to be applied to the artifacts and products developed.

As it is configurable, the CCP can be tailored according to the different management needs of different types of projects. For example, a research project may allow unlimited access to all resources by all members of the small group of professional staff assigned to do the research, but a project to develop a million lines of mission critical code for the Space Station that has 100 programmers working simultaneously would probably need to strictly control, not only access, but also the tools to be used for each step of the process.

Today, the CCP is still in the design phase. It will utilize both conventional and knowledge-based technology; a knowledge-based system is planned to help configure the CCP. The IDEF3 method for process description, a graphical method developed for the U. S. Air Force, will be used to specify the steps in the software development process. The user interface will display these IDEF3 process diagrams as a guide for the user; shading or coloring will be used to indicate which activities have been completed and which ones the user is not authorized to perform (and will not be allowed to access). Buttons and menus will be used to help choose the best tool(s) to perform each task and to list the artifacts which must be produced before the task is considered complete. The user interface will also display a matrix of software development perspectives versus task focus, based upon concepts of Information Systems Architecture suggested by J. Zachman [23].

6. COST MODELING

Organizations such as NASA that make large expenditures for the development of software can realize significant benefits by including a good model for the estimation of software costs in their set of software engineering tools. COSTMODL was designed to be an in-house tool for the estimation of NASA software development costs, but it can be used by other organizations as well [24]. It provides project managers with an automated tool that permits them to obtain credible development cost estimations without having to rely on software costing specialists and to check estimates provided in contract proposals.

6.1 Characteristics of COSTMODL

COSTMODL is a flexible tool for estimating the effort and schedule required to develop a software product of a given size. Five separate cost estimation models have been incorporated into

COSTMODL. These are the KISS (Keep It Simple Stupid) model, the Basic, Intermediate and Ada COCOMO models, and the Incremental Development Model. This choice of models covers the spectrum of project complexities from small, relatively simple projects to very large projects developed in an Ada software engineering environment and delivered in a series of separate but related increments. All of the parameters defining each of the models are accessible to the user. The basic estimating equations can be calibrated to the user's software development environment and type of products, and the set of factors which influence software development costs can be redefined.

6.2 Definition of the Models Used

KISS. The KISS model is a simplified linear estimating model which was developed using productivity data derived from past NASA software development projects. In addition to the anticipated product size, the user specifies the project criticality, type of software, type of language, development team productivity and an Ada Productivity factor.

COCOMO. The COCOMO model, originated by Dr. Barry Boehm, is the most widely used cost estimation model due to its proven performance over the years, and to the fact that it is in the public domain, allowing one to tailor it to its own environment.It consists of a family of models.
The Basic COCOMO model is a very simple approximation which accepts only the size of the total product to perform the evaluation. It is good for quick early rough order of magnitude estimates.
The Intermediate COCOMO model is a more sophisticated model which provides a mechanism whereby the user can specify a set of factors to account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques and other project attributes known to have a significant influence on software costs.
The Ada COCOMO model is an extension to the Intermediate model which attempts to quantify the changes in programmer productivity resulting from the use of good software engineering practices.
Incremental Development model. The incremental development model provides for the division of a total project into separate stand-alone deliveries. It computes the individual increment effort and schedule and the total project effort and schedule, taking into account the amount of rework required on the earlier increments to accommodate the later increments.

6.3 Status

COSTMODL is now distributed for NASA by Computer Sciences Corporation (713-280-2233) and is being prepared for distribution through the Computer Software Management and Information Center (COSMIC) at the University of Georgia. It is available free to government agencies and their contractors. It is being used at several hundred government, military and contractor sites, and has been selected as the standard cost estimating tool for NASA's Space Station Freedom Program (SSFP).

7. TECHNOLOGY TRANSFER

Technology transfer of emerging CASE technology to NASA is the primary objective of all of the STB's CASE projects. Technology transfer can occur in many different ways, including the direct use of tools developed by the STB (such as COSTMODL), the adoption of new methods and technology identified by the STB (such as repository technology when it matures), and the purchase of appropriate commercial off-the-shelf (COTS) tools. This section discusses a process being developed by the STB to help insert appropriate CASE technology at JSC.

What techniques should be used to select and insert CASE technology? At first, the STB set a goal of collecting information about existing CASE products and characterizing those products in order to provide a CASE tool consulting service for the JSC community. The motivation for establishing such a service was the complex nature of CASE and the confusing status of the CASE tool market. The sheer number of available CASE tools and the rapid rate of change of the CASE market, coupled with unrealistic consumer expectations of what CASE tools can do, have led to some exaggerated claims about CASE and, consequently, to some disappointed consumers.

Rather quickly, the scope of this CASE consulting service was expanded to a software engineering consulting service. This is because CASE tools cannot produce magical results (despite the claims of many vendors); i.e., CASE tools can only *assist* in the software engineering process. A software development organization must still employ people to do good software engineering and must still have a well-managed, repeatable, and *explicit* software development process. If a disciplined software engineering process does not exist within an organization, then that organization must adopt one, and this will likely imply a change in its way of doing business.

How do organizations successfully introduce new ways of doing business? Technology insertion, in particular, seems to be successful only when key people within the organization (called change agents) actively participate. Change agents tend to be knowledgeable persons in middle management, recognized for their abilities and credibility by both upper and lower management.

In order to make sound CASE recommendations and to improve the chances of achieving CASE technology insertion, the STB is developing a CASE Selection and Insertion Process [25,26]. In simplest terms, there are five basic activities that occur during the process: characterize the organization's culture; characterize the software systems produced; identify improvements to the organization's software engineering process; identify candidate tools and environments; and develop a technology insertion plan.

As a consequence of an organization's request to the STB for assistance in the choice of automation techniques for their software development, there is an initial survey of the situation and problems faced by the organization. Meetings are held with managers who identify persons to be interviewed for information and to support the process.

The characterization of the "culture" of the organization has some similarity to the Software Engineering Institute's Software Maturity Assessment. It is important to have an in-depth understanding of how the organization currently does business, i.e., what the characteristics of the software development process in use by the organization are, and which software engineering methods and notations are already understood or preferred. If there is no documented process nor consistent methods and if there is no tool which supports the current way the organization works, a lot of change is in order.

The type of applications being developed by the organization must be determined. This is important because different CASE tools are required for different types of applications. In the simplest model, all applications have three components to them: data, function, and control. An MIS system is data-focused, and so the data modeling capabilities of the CASE tool are of primary importance. On the other hand, an engineering application with real-time characteristics has design concerns primarily with function and control, and so the CASE tool must be able to model the behavior of the system appropriately. Some other important characteristics are the database and language to be used in the applications being developed, especially if the CASE tool will be required to do code generation.

The identification of possible improvements to the organization's software engineering process is a critical portion of the technology insertion work. Potential improvements should be frequently discussed with the organization and feedback obtained. Feedback from the organization is essential because the STB is trying to provide useful advice that will solve some of the organization's real needs and is not trying to dictate policy.

Once the organization and its applications are characterized and analyzed, a mapping is made between these results and the characteristics of CASE tools. A set of tentative requirements emerges which is used to filter potential candidate tools from the universe of known tools. The weighting factors that should be applied to particular requirements are determined. These weights are affected by trade-off considerations such as the long-term benefits versus coststo the software development organization to change its current process or methods or to obtain different or upgraded hardware, if necessary.

The universe of known tools that is referenced in order to identify candidate tools consists of three collections. One is a manual file collection of some 90 vendors' brochures, demonstration disks, and commentary thereon kept in file drawers. A second is a public domain database from the Air Force's Software Technology Support Center (STSC) which contains commentary on tools tested by or used by contributors to the database. (Users are expected to update the database with local commentary and return it to STSC on a quarterly basis.) The third collection of information is a commercial database from P-Cube, Inc. which allows the user to query for tools on certain prescribed capabilities. (P-Cube sends monthly updates.) The electronic databases are maintained in the STB's Laboratory.

Armed with a short list of candidate tools, the STB presents its findings to the software development organization, obtains additional feedback, and repeats the requirements definition and tool filtering as many times as is appropriate. However, the organization must obtain hands-on experience with the candidates prior to *their own* final selection. To conclude the consulting process, the STB presents the completed technology insertion plan. The STB provides any final suggestions for improvements to the software engineering process and recommends the necessary training for the software engineering methods to be used, the new hardware, and the CASE tool itself. CASE tools that have extensive capabilities for use with complex systems are non-trivial to use. To avoid failure, training for new users and a continuing program of training are essential.

Currently, the CASE Selection and Insertion Process is being applied within two other branches of the Information Systems Directorate at JSC. Support for MIS applications has been targeted for this initial testing of the Process. Preliminary findings, recommendations, and technology insertion plans have been presented to these two organizations. The STB is currently reviewing the results of this initial testing of the Process in order to revise it accordingly. Discussions are also in progress between the Mission Operations Directorate and the STB regarding the need for CASE support.

8. PLANS FOR THE FUTURE

The STB will continue to track the progress of software engineering technology, especially in CASE environment frameworks, standards, and methods. Development and refinement of the REAP environment, the ASDW technology and tools, and COSTMODL is continuing. A strong synergistic potential exists between the REAP and ASDW projects, especially in the area of domain-specific architectures, a research area aimed at identifying standard architectures for various domains in order to permit the development (or reengineering) of standard, reusable parts for each of these architectures. The potential for collaboration also exists between the ASDW project and two other STB projects, the Intelligent Computer-Aided Training (ICAT) project and the Task Analysis/Rule GEnerating Tool (TARGET) knowledge acquisition project. These areas of cooperation will be actively pursued. Finally, the CASE Selection and Insertion Process will be refined and extended, and the STB will examine the feasibility of setting up a CASE product demonstration facility to provide an opportunity for potential users to assess the features of a choice of CASE tools and environments.

9. SUMMARY AND CONCLUSION

This paper has discussed ongoing work within the Software Technology Branch (STB) to identify and develop some of the Computer-Aided Software Engineering (CASE) technology that NASA will need to meet the software development and maintenance challenges of the future. The main goals of this work are to improve the productivity of software developers, users, and maintainers and to improve the quality of software systems. Specifically, the following STB projects have been discussed in this paper:

- The STB has identified, evaluated, and characterized leading CASE products, and projections of the long-term prospects and direction of CASE have also been made. (Sec. 3)
- The Reengineering Environment Application Project (REAP) has proposed Fortran standards, reengineering methods, and the types of tools required in an ideal, integrated, reengineering tool set. A preliminary environment to support maintenance and reengineering analysis has been developed. (Sec. 4)
- The Advanced Software Development Workstation (ASDW) project is researching and developing specific types of advanced technology and tools that an advanced workstation for software development should provide. A Parts Composition System (PCS), which includes an Engineering Script Language (ESL) editor, for developing applications from reusable software parts is being developed. Testing of a PCS subshell, the INTelligent User Interface development Tool (INTUIT), has been recently completed. The Framework Programmable Platform (FPP) subtask is developing a Configurable Control Panel (CCP) for an integrated CASE environment, which will enforce an organization-specified model of the software engineering process. (Sec. 5)
- COSTMODL is a mature, flexible tool for estimating the effort and schedule required to develop a software product of a given size. It is currently being used at several hundred government and contractor sites.(Sec.6)
- Technology transfer of emerging CASE technology to NASA is the primary objective of all of the STB's CASE projects. The STB has developed a CASE Selection and Insertion Process and is currently testing it. (Sec. 7)

In order to meet the challenges of tomorrow's space program, there is a need to increase the software process maturity level of NASA organizations and contractors and to enforce good software engineering principles. The STB believes that the use of CASE tools will be a major benefit to NASA in the coming years. It is anticipated that the technology being identified and developed by the STB, and by many other NASA organizations as well, will play a strategic role in future NASA software development projects.

10. ACKNOWLEDGMENT

We acknowledge the National Research Council's sponsorship (through its Research Associateship Program at the Johnson Space Center) of M. E. Izygon, one of the authors of this paper.

11. REFERENCES

1. Earl, Anthony, "A Reference Model for Computer Assisted Software Engineering Environment Frameworks," Ver. 4.0 ECMA/TC33/TGRM/90/016, Hewlett-Packard Labs., Software Environments Group, Bristol BS12 6QZ, England, Aug., 1990.

2. Simmons, G. L., "What is Software Engineering?" ISBN 0-85012-612-6, NCC Publications, 1987, as quoted in Earl [1], p. 1.

3. Boehm, Barry W., "The Goals of Software Engineering," SOFTWARE ENGINEERING ECONOMICS, Prentice-Hall, Englewood Cliffs, NJ, 1981, p. 23.

4. Sommerville, Ian, "Introduction," SOFTWARE ENGINEERING, 3rd Ed., Addison-Wesley, Menlo Park, CA, 1989, p. 20.

5. Boehm [3], p. 16.

6. NASA, Office of Safety, Reliability, Maintainability, and Quality Assurance, Software Management and Assurance Program (SMAP), "NASA Software Acquisition Life Cycle," Ver. 4.0, NASA, Washington, D. C., 1989.

7. Booch, Grady, "The Software Crisis," and "Software Engineering," SOFTWARE ENGINEERING WITH ADA, 2nd Ed., Benjamin/Cummings, Menlo Park, CA, 1987, pp. 8 &29.

8. Booch [7], pp. 36-37.

9. Humphrey, Watts, "CASE Planning and the Software Process," CMU/SEI-89-TR-26, Carnegie-Mellon University/Software Engineering Institute, Pittsburgh, PA, 1989.

10. Rogers, Kathy L., "Software Engineering Environment Frameworks, Vol. I: Evaluation Method," MTR-91W00048-01, The MITRE Corp., Houston, TX, April, 1991.

11. Rogers, Kathy L., "Software Engineering Environment Frameworks, Vol. II: Evaluation of the Software Life Cycle Support Environment," MTR-91W00048-02, The MITRE Corp., Houston, TX, May, 1991.

12. Fridge III, Ernest, Braley, Dennis, & Plumb, Allan, "Maintenance Strategies for Design Recovery and Reengineering," Vols. 1-4, NASA Johnson Space Center, Houston, TX, June, 1990.

13. Plumb, Allan, & George, Vivian, "A Method for Conversion of Fortran Programs," Barrios Technology, Inc., Houston, TX, March, 1990.

14. Braley, Dennis, "Automated Software Documentation Techniques," NASA Johnson Space Center, Houston, TX, April, 1986.

15. Braley, Dennis, "Software Development and Maintenance Aids Catalog," JSC-22349 (86-FM-27), NASA Johnson Space Center, Houston, TX, October, 1986.

16. Fridge III, E. M. and Pitman, C. L., "The Advanced Software Development Workstation Project," SOAR90, Albuquerque, NM, June 26-28, 1990.

17. Pitman, C.L., Izygon, M.E., Ralston, E.W., FridgeIII, E.M., and Allen, B.P., "Intelligent Interfaces For Complex Software," Fourth International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, Kauai, Hawaii, June 2-5, 1991.

18. NASA Goddard Space Flight Center, "Transportable Applications Environment Plus," Ver. 4.1, GSC-13275 with documentation, COSMIC, The University of Georgia, Athens, GA, Jan., 1990. Ver. 5.1 (MOTIF), GSC-13448, released May, 1991.

19. Szczur, Marti, "Transportable Applications Environment (TAE) Plus: A NASA Tool Used To Develop And Manage Graphical User Interfaces," SOAR91, Houston, TX, July 9-11, 1991.

20. UNISYS Houston Operations, GNDSIM User's Guide.

21. Izygon, M.E., and Pitman, C.L., "A Knowledge Based Front-end for a Complex Space Flight Simulation Program," 1991 Summer Computer Simulation Conference, Baltimore, MD, July 22-24, 1991.

22. Mayer, R. J., Blinn, T. M., and Mayer, P. S. D., "Framework Programmable Platform for the Advanced Software Development Workstation: Concept of Operations Document," Report to NASA and University of Houston-Clear Lake by Knowledge Based Systems Inc. under subcontract SE.37, NCC9-16, Sept. 18, 1990.

23. Zachman, J., "A Framework For Information Systems Architecture," IBM Systems Journal., Armonk, NY, Vol. 26., No. 3, Sept., 1987, pp. 276-292.

24. Roush, G. B., "COSTMODL User's Guide," Preliminary Draft, NASA, Johnson Space Center, Software Technology Branch/PT4, Houston, TX, March, 1991.

25. Erb, D. M., "Evaluation and Selection of CASE Products," Presentation to Software Technology Branch, The MITRE Corp., Houston, TX, Nov. 19, 1990.

26. Pitman, C., Erb, D., Rogers, K., and Dorofee, A., "The CASE Selection Process: Essential Data and Conceptual Example," Ver. 1.0, NASA, Johnson Space Center, Software Technology Branch/PT4, Houston, TX, May 16, 1991.

27. Open Systems Foundation (OSF), "OSF User Environment Component: Decision Rationale Document," OSF, Jan. 11, 1989, as quoted in [1], pp. 51-52.

APPENDIX

This appendix gives a brief description of each of the major groups of services in the NIST/ECMA reference model [1] shown in figure 1.

Data Repository Services. The maintenance, management, and naming of data entities or objects and the relationships among them is the general purpose of the data repository. Basic support for process execution and control is also addressed here along with a location service to support physical distribution of data and processes. The classes of services in this group are: data storage, relationship, name, location, data transaction, concurrency, process support, archive, and backup.

Data Integration Services. The data integration services enhance the data repository services by providing higher-level semantics and operations with which to handle the data stored in the repository. The classes of services in this group are: version, configuration, query, metadata, state monitoring, sub-environment, and data interchange.

Tools. The entire set of environment framework services exist partly to support one another, but mainly to support tools that provide assistance for particular forms and methods of software engineering. Tools plug into the CASE environment framework. In addition to fully integrated tools that extensively use an environment framework's services, the tool slots in an environment framework may also provide an **encapsulation service.** This service is used when a tool exists (but was not written to make use of any of a particular environment framework's services) and is made to work in the environment framework by surrounding the tool with software that acts as a layer between the tool and the framework. The encapsulated tool fits into the framework without modification, but it uses very few (if any) of the framework's services.

Task Management Services. These services, sometimes called **software process management services,** allow the user to deal with major tasks as opposed to accomplishing each job by a tedious series of invocations on individual tools. The classes of services in this group are: task definition, task execution, task transaction, task history, event monitoring, audit and accounting, and role management.

Message Services. These services provide standard, two-way communications between services, between tools, and between tools and services. The classes of services in this group are: message delivery and tool registration.

User Interface Services. The importance of separating the presentation of functionality from the provision of functionality is widely recognized, and a consistent user interface service may be adopted for a complete environment framework. However, because of the complexity and generality of user interface issues, the NIST/ECMA model does not include a user-interface reference model of its own. It does summarize an existing user-interface reference model, which it recommends as a good starting point for discussions and which may be described as the (relatively) familiar "X-Windows layered model" [27].

Security Services and Framework Administration and Configuration Services. These last two groups of services are not shown in figure 1. Security services affect all of the other groups and cross many of the logical group boundaries represented in figure 1. Three classes of security services are: security information, security control, and security monitoring. Framework administration and configuration services are vital to a CASE environment, but the reference model (conceptually) views these services as customizations of the other framework services already discussed.

The NIST/ECMA reference model also specifies a way to completely describe each of the services mentioned above, in order to ensure compatibility, comparability, and precision of descriptions. In their presentation of the reference model, the following **dimensions** are used as sub-headings for the descriptions of particular services: Justification, Conceptual, Degree of Understanding, Data dealt with, Types, Operations, Rules, Metadata, Instances, External, Internal, and Related Services [1]. The scope of this paper does not permit further discussion of these dimensions.

SYSTEM DIAGNOSTIC BUILDER

Joseph L. Nieten
GHG Corporation
1300 Hercules, Suite 111, Houston, TX 77058

Roger Burke
NASA,JSC
DK42, Houston, TX 77058

ABSTRACT

The System Diagnostic Builder (SDB) is an automated software verification and validation tool using state-of-the-art Artificial Intelligence (AI) technologies. Originally developed by GHG Corporation of Houston, Texas, the SDB is used extensively by project BURKE at NASA-JSC as one component of a software re-engineering toolkit. The SDB is applicable to any government or commercial organization which performs verification and validation tasks.

The SDB has an X-window interface, which allows the user to 'train' a set of rules for use in a rule-based evaluator. The interface has a window that allows the user to plot up to five data parameters (attributes) at a time. Using these plots and a mouse, the user can identify and classify a particular behavior of the subject software. Once the user has identified the general behavior pattern of the software, he can train a set of rules to represent his knowledge of that behavior.

The training process builds rules and fuzzy sets to use in the evaluator. These rules are built using a special implementation of the ID3 algorithm. The fuzzy sets classify those data points not clearly identified as a particular classification. Once an initial set of rules is trained, each additional data set given to the SDB will be used by a machine learning mechanism to refine the rules and fuzzy sets. This is a passive process and, therefore, it does not require any additional operator time.

The evaluation component of the SDB can be used to validate a single software system using some number of different data sets, such as a simulator. Moreover, it can be used to validate software systems which have been re-engineered from one language and design methodology to a totally new implementation.

Theory of Operation

The System Diagnostic Builder (SDB) uses an inductive machine learning technique to generate decision trees from data sets classified by a Subject Matter Expert (SME).

The primary objective of the SDB is to capture system knowledge from an SME. This process is known as knowledge acquisition. The knowledge must be represented in a manner to maximize usability. Representing the knowledge with a rule-base enables it to be used by a standard expert system shell or any other rule-based system. These rules could be used by a forward chaining system to detect faults and/or by a backward chaining system to identify the causes of these faults.

The SDB knowledge acquisition process is based on machine learning techniques combined with an X-window graphical interface. This interface allows an SME to train a rule-base from some number of data sets. The SME merely selects those parameters deemed to be pertinent to the identification of a particular system state. These parameters are then plotted on the screen, and with a click of the mouse button, the SME assigns some number of those data instances to a system state. This process is repeated until the entire data set is classified.

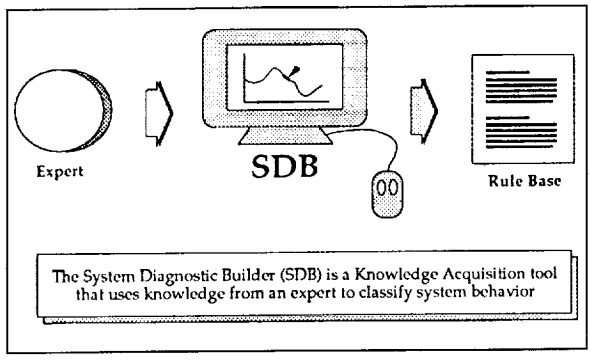


Figure 1

Those parameters used by the SME to classify system behavior must be available from some data stream within the system. However, the SME may require additional parameters during the classification process to insure accurate identification of the system's behaviors. These additional parameters will depend in some way on those basic parameters available from the system's data stream. The knowledge about these new parameters represents a whole new area within this knowledge acquisition scheme.

Additional knowledge about derived parameters is supplied to the SDB through the use of a standard ASCII text file which defines the relationships between actual system parameters and those parameters derived from the actual parameters. Future plans for the SDB include the incorporation of a prolog type dialog system to extract this Meta knowledge from the SME.

A temperature parameter is one example of a basic parameter used for training. Alone this parameter may not indicate very much, however, when used to calculate another key parameter, it becomes very useful. The knowledge about these calculations can either be generic to the type of science used by the system or explicitly added by the SME.

Once a data set is classified to the best of the SME's knowledge, it is submitted to an induction routine for generation of the rule-base. GHG has developed a special induction algorithm to generate the required rule-base. This new algorithm, called Turbo Induction, is a special implementation of the ID3 algorithm. In general, Turbo Induction enhances the discrete logic capabilities of ID3 with functions that map the continuous values of the system's parameters into discrete events. Specific capabilities of Turbo Induction are described later in this paper.

The SDB has an incremental learning capability. Multiple SME's can train the same data set, or a single SME can train multiple data sets without performing negative training. This is accomplished by managing the rule-bases from the X interface and by eliminating that SME knowledge which causes a conflict with existing knowledge. At this point, this process is done without consulting the SME. Future versions will provide a dialog capability to inform the SME of existing conflicts in the training domain for that system and request clarification from a 'super' SME.

Turbo Induction

Induction is a process where rules are automatically generated from a set of examples. ID3 is an induction algorithm developed by J.R. Quinlan. This algorithm analyzes data sets and recursively creates a decision tree. Each pass through the algorithm, a single attribute to be analyzed is chosen and a breakpoint value is calculated. This breakpoint is the largest inflection point. A node is added to the decision tree identifying the values on both sides of the breakpoint.

This approach depends heavily on the choice of attribute at each pass. A bad choice of attributes yields a set of rules that are unnecessarily complex. This approach also assumes that optimal decisions are made from existing data. There are no provisions for data provided by an SME during the operation.

While the ID3 algorithm works very well for discrete parameters and data, current implementations do not extend that success into the domain of continuous functions. Extremely large and noisy data sets produce a computational nightmare. For this reason, GHG Corporation has developed it's own implementation of the ID3 algorithm, with specific emphasis on training rules from very large and noisy data sets. This implementation is called Turbo Induction.

Several methods have been researched to eliminate the effects of noise on the induction process. One method took random samples from the data set and performed induction on only those data points. This method became less effective as the size of the data set grew. Analysis revealed that the density of those values which indicated particular paths through the decision tree became saturated, and actually diluted the other paths having less data points to support them. This is unacceptable when training a validation system.

Turbo Induction analyzes data sets using a pre-processor methodology. It performs localized induction and then maps those results into a final iteration that yields a set of discrete events. Thus, mapping the values from continuous functions into some number of discrete states (membership in a set). These sets are constructed and populated based on all knowledge about the system.

Turbo Induction has a procedure to add those parameters deemed important to the construction of a reliable rule-set. A superset of additional parameters must first be created. These parameters are analyzed during the pre-processor phase to determine their contribution to the final decision tree.

Applications

Currently, Project BURKE at NASA-JSC is using the SDB to Verify and Validate (V&V) the Shuttle Mission Simulator (SMS).

The SDB is used to train rule-bases for each single system within the SMS. Each of these systems has it's own data parameters to indicate it's behavior. An SME from each system trains a set of rules using data collected from the SMS. The SME also trains a set of rules using data collected from an actual flight or from another simulation which has already been certified (some kind of baseline).

Each of these rules is tested to determine it's accuracy level. In the event the generated rules are not accurate enough, the SME has the ability to refine the training and regenerate the rules. Once a valid set of rules exists for both the SMS and the baseline system, the sets of rules are combined to form a rule-base for the subject system.

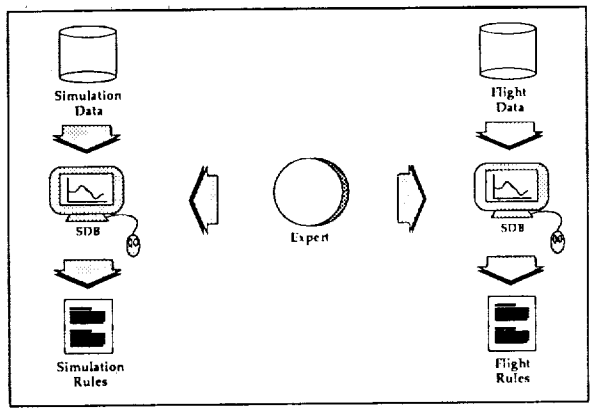


Figure 2

The verification and validation process is done passively. An operator can start a background process to observe the SMS data stream and classify the behavior of each data instance in real-time. This mode does not require an operator to monitor progress, only to respond to the warnings. The operator can also use data files to compare behaviors off-line. This approach can take more man power.

In short, an expert system shell is using the generated rule-base to identify the current state of the subject system. As each data instance is presented to the expert system, two rules should fire: One rule from the SMS rule set and one rule from the baseline rule set. The conclusion of both of these rules should agree. If they do not agree, then the expert system has identified an inconsistency.

Inconsistencies can be attributed to several reasons. The rule-sets may not agree due to an actual problem occurring in the baseline; which was then represented in the baseline rules set. In this case, the inconsistency is attributed to an anomaly that was not incorporated into the simulation. However, an inconsistency can also be an indication that the simulation has not modeled the real world properly. This situation is of particular interest for this application.

The SDB can be used in a number of Knowledge-Based applications. Each of these is discussed below.

As demonstrated in the SMS application, the SDB can be used to compare a software system's behavior to the behavior of some baseline system. The only requirement to perform this kind of analysis is data; data from both the software system to be analyzed and the baseline system.

Using the SDB to V&V re-engineered code is conceptually the same as the V&V process of a simulator. The only difference is the source of the baseline used for comparison. A subject for the baseline is already available: the original program source code. This makes the use of the SDB a perfect fit. The only difficulty may be in the visibility of variables within the original software.

Other applications can make use of the rule-bases generated by the SDB. The rule-base generated by the SDB represents actual 'behavioral' knowledge about the subject system. This knowledge is portable to any other rule-based system. In particular, these rules can be used in fault detection systems, fault isolation systems, and Intelligent Computer Aided Training (ICAT) systems.

One of the biggest draw backs for conventional expert systems is the time required to extract the knowledge from the expert. The SDB provides a vehicle for system knowledge to be captured one time and reused by any rule-based application.

Conclusion

The SDB provides a unique tool to perform knowledge acquisition for those systems with accessible data. The SDB also provides an excellent platform to perform verification and validation of conventional systems, using state-of-the-art technology.

REFERENCE

Holland, J.H., "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems", in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Quinlan, J.R., , "The Effect of Noise on Concept Learning," in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Quinlan, J.R., "Induction of decision trees", *Machine Learning 1*, 1.

Quinlan, J.R., "A case study of inductive knowledge acquisition", in *Applications of Expert Systems*, Quinlan J.R., Addison-Wesley.

Quinlan, J.R., "Generating production rules from decision trees", Proceedings 10th International Joint Conference Artificial Intelligence, Milan.

O'Keefe, R., "Simulation and Expert Systems - A Taxonomy and Some Examples", in *Simulation*, Society for Computer Simulation San Diego.

N93-11945

KNOWLEDGE-BASED SYSTEM V&V IN THE SPACE STATION FREEDOM PROGRAM

Keith Kelley & David Hamilton
IBM Federal Sector Division
MC 6402
3700 Bay Area Boulevard
Houston, Texas 77058

Chris Culbert
NASA/Johnson Space Center
Software Technology Branch/PT4
Houston, Texas 77058

ABSTRACT

Knowledge Based Systems (KBSs) are expected to be heavily used in the Space Station Freedom Program (SSFP). Although SSFP Verification and Validation (V&V) requirements are based on the latest state-of-the-practice in software engineering technology, they may be insufficient for Knowledge Based Systems (KBSs); it is widely stated that there are differences in both approach and execution between KBS V&V and conventional software V&V. In order to better understand this issue, we have surveyed and/or interviewed developers from sixty expert system projects in order to understand the differences and difficulties in KBS V&V. We have used this survey results to analyze the SSFP V&V requirements for conventional software in order to determine which specific requirements are inappropriate for KBS V&V and why they are inappropriate. Further work will result in a set of recommendations that can be used either as guidelines for applying conventional software V&V requirements to KBSs or as modifications to extend the existing SSFP conventional software V&V requirements to include KBS requirements. The results of this work are significant to many projects, in addition to SSFP, which will involve KBSs.

INTRODUCTION

Knowledge-based systems, or expert systems, are in general use in a wide variety of domains. (Although there is a growing acceptance of different definitions for knowledge-based systems and expert systems, we will use the terms interchangeably in this paper. The differences between KBS and expert systems do not significantly affect the V&V process.) As reliance on these types of systems grows, the need to assess their quality and validity reaches critical importance. As with any software, the reliability of a KBS can be directly attributed to the application of disciplined programming and testing practices throughout the life-cycle. However, there are essential differences between conventional software and knowledge-based systems, both in construction and use. The identification of how these differences affect the verification and validation (V&V) process and the development of techniques to handle them is the basis of work in this field.

Much of the work in KBS V&V has focused on developing conceptual approaches and postulating different techniques for performing some or all aspects of V&V on various types of KBSs or expert systems (ESs) [5]. Very little work in this field has demonstrated the usefulness of proposed techniques on operational KBS. Even more importantly, since effective V&V must be applied throughout the life-cycle, there has been almost no case study work in applying disciplined software V&V principles throughout the development of an operational KBS. The long term goal of our work is to develop guidelines, standards, tools, and techniques for V&V of all KBS applications which many be used in the Space Station Freedom Program (SSFP). As a precursor to determining the applicability or usefulness of many of the proposed KBS V&V techniques, it is important to develop an understanding of what V&V practices are commonly in use today and how proposed techniques can improve upon those practices.

It has been widely claimed that few expert systems are subjected to the same level of V&V that conventional

software routinely undergoes [4]. However, this practice has not been well documented. More important for our purposes, little documentation exists (an exception is documented in [8]) which describe the problems associated with KBS V&V from the developer or user's point of view. The specific purpose of our survey was to begin documenting the experiences and problems KBS developers have encountered in performing V&V on their systems and relate those problems to the kinds of issues KBS V&V researchers consider important. The overall strategy for determining the state-of-the-practice was to determine how well each of the potential expert system V&V issues are being addressed and to what extent they have impacted the development of expert systems. Our approach was to develop a set of survey questions for both KBS developers and users and then to follow that survey with selected interviews.

Because our ultimate goal is to develop guidelines, etc. for SSFP, we compared the results of our survey to the existing SSFP V&V requirements. We also analyzed all the SSFP V&V requirements to determine their general applicability to KBS V&V.

In this paper, we first summarize the results of this survey (a more complete discussion of the survey results appears in [9]) and then we summarize the results of analyzing SSFP V&V requirements.

SURVEY RESULTS

A total of 70 people, 93% of which were developers, responded to the survey concerning a variety of knowledge-based systems. Seventy percent of these systems were operational and the remainder were considered prototypes (although some of these "prototypes" had users). These systems covered a range of criticalities and sizes, requiring as little as one person-month of development effort to as much as two hundred person-months of development. Most (75%) of the systems were concerned with diagnosis, primarily in the aerospace field (73%).

Questionnaire Results

Much of the results can be derived by simply calculating the fraction of respondents that answered a question in a certain way. The following is a short summary of each type of information gathered. Unless otherwise noted, the percentages shown are the percentage for all the responses, both developer and user combined.

Performance Criteria

Thirty-nine percent estimated that the expert system performed with an actual accuracy of less than 90% and 54% estimated an accuracy of less than 95%. Most (50%) estimated the problem space coverage between 60% and 95%. In comparing the accuracy of the expert and the expert system, most (79%) expected the expert system to at least as accurate as the expert. Yet, the actual systems were often (75%) estimated to be less accurate than expected and also (62%) less accurate than the expert. Users, more often than developers, estimated the expert system as being less accurate than expected and less accurate than the expert.

Requirements Definition

Seventy-five percent indicated that expert consultation was a basis for determining the behavior of the system. More revealing is that for 52% of the systems surveyed, there were no documented requirements. Forty-three percent indicated that prototypes or similar tools were used for requirements. Forty percent had medium difficulty in generating requirements, 35% said the requirements were hard to develop, 25% said the requirements were easy to develop. Fifty-eight percent of developers had a high level of contact with experts during development.

Development Information

The most frequent (40%) life-cycle model used is the Cyclic Model (repetition of Requirements, Design, Rule Generation, and Prototyping until done). However, 22% of the respondents stated that no model was followed. Most development was done with an expert system shell (CLIPS and others), and the predominant Interface Code was C and LISP. Applications were reasonably large, requiring an average of 23 person-months to develop. Developed systems were not reported to be particularly sensitive to change (77% said changes only occasionally caused an unexpected behavior).

V&V Activities Performed

Most V&V activities relied on comparison with expected results and checking by the expert. Sixty-six percent used functional testing and 44% used structural testing. Fifty-nine percent had the domain expert check the knowledge base. On average, 24% of the development was spent on V&V. While all (100%) of the users rated V&V of expert systems as hard, the response from developers varied. Thirty-four percent of the developers said the V&V effort was of medium difficulty while 27% said it was hard and 33% said it was easy, 5% said it was impossible. Significantly, each V&V technique was used as the sole V&V technique in at least one project. Also, in general, there were wide ranging uses of V&V techniques; each technique was used by many projects.

V&V Issues Encountered

The known issues most often cited as problems were: test coverage determination (63%), knowledge validation (60%), real-time performance analysis (33%), and

problem complexity (40%). Other problems cited were: modularity (27%), configuration management (20%), certification (11%), and understandability (10%). The least cited problem was analysis of certainty factors (only seven respondents indicated that certainty factors were used). Every known issue was cited by at least one respondent. The expected system use varied widely (3-2000), while actual system use was relatively good. However, less than half of the respondents provided information, suggesting that actual use was much lower than reported. Of those who responded with an opinion, 96% felt that their expert system was at least as reliable as a typical conventional software system, and 51% felt it was more reliable.

Interview Results

In addition to acquiring written responses to the survey questions, interviews were performed to gather additional data and to clarify questions concerning the written responses. Additional information from these interviews are summarized in this section.

Structural Testing

Based on the survey results, a commonly used evaluation approach was the use of structural testing. This was surprising because the common perception among KBS researchers is that many common forms of structural testing are relatively difficult to apply to expert systems. From the interviews, we learned that although some projects did attempt to measure the actual test coverage (i.e., percentage of rules executed during testing) many others did not actually measure the coverage. Instead, they attempted to develop test cases that would cover all of the knowledge base (or at least the important parts) but made no attempt to measure how well the knowledge base was actually covered. Also, there appeared to be no attempt to cover interactions between knowledge base elements (e.g., rule interactions). Generally, each element was tested as if it were an independent piece of the knowledge base. Some knowledge base developers felt that more formal structural testing would be too much effort and would hinder the development process too much. The interview results suggest that although structural testing was used, it was a very weak form of structural testing (at least compared to, say, branch coverage in procedural software testing).

Experts Developing Expert Systems

It appeared that the expert was heavily relied upon to aid in evaluation of the knowledge base; this subject was probed more deeply during the interviews. The developers felt that a close interaction between the expert and the knowledge base developer was mandatory to successfully develop an expert system. This is not a surprising result and it has been discussed at length in the literature [1]. Many KBS developers feel this interaction is so important that they think the best approach is simply to have the expert develop the system. Though it is important for a knowledge engineer to understand the problem domain and to thoroughly represent that domain [6], it is generally accepted that the domain expert should not be the sole developer of an expert system: this is described in more detail in [7], p.154 as the Knowledge Engineering Paradox: "The more competent domain experts become, the less able they are to describe the knowledge the use to solve problems." There are many problems associated with the development of an expert system by a domain expert. Experts often use knowledge that is so highly compiled and implicit that they have difficulty defining that knowledge explicitly (so a machine can use it). Furthermore, collection of domain knowledge from "introspection" is generally held in doubt by psychologists [3]; that is, experts often don't solve a problem the way that they think they do. Finally, building expert systems often involves building highly complex software systems, systems that require skills and training that domain experts seldom have. Some of these issues were recognized by at least one interviewee who felt that when his group begins to tackle more sophisticated problems, they would need developers with better-developed software and knowledge engineering skills.

Requirements Writing and the Conventional Software Life-Cycle

We anticipated that expert systems were being developed using a much more iterative and less structured life-cycle than the conventional waterfall model. Although the subject of life-cycle models was not intentionally addressed during the interviews, it often came up when discussing requirements. It seems that several respondents associated "requirements" with the conventional waterfall model. They felt very strongly that the conventional approaches to software development, such as the waterfall model, were much too formal and structured for expert systems development. Some even suggested it would be disastrous to apply them to expert systems. For many, this feeling extended to documenting requirements, others simply used a different approach to requirements. For example, in some cases, requirements were not written because it was felt that a requirements document was a formally written paper document that needed to be "approved" before development could proceed. In other cases, an iterative prototyping development effort took place and was followed by documenting system requirements. These requirements were then used to test the system to ensure that it worked as everyone thought it should.

Prototypes vs. Operational Systems

Although we asked respondents to state that their system was either "a prototype" or "operational," we received

indications that this distinction was often difficult to make. For example, responses included "it is both a prototype and operational," or "it is an operational prototype," or "it is just a prototype but we have many users." It seems that some systems are originally intended to be a prototype but are used operationally. Some intentionally approach the development of an operational system by first developing a "prototype" and once the prototype is "certified," it is considered "operational." Others acknowledge there is a danger that a prototype will be used as if it were operational. They have taken steps to ensure that a prototype system that is not accidentally relied upon in an operational setting.

Real-Time Performance Analysis

In our survey, we intended "real-time performance analysis" to refer to the ability to predict the response time for an expert system. That is, the ability to analyze the time performance of the system. However, from the interviews we learned that many interpreted "real-time performance analysis" to mean the ability to get the system to run as fast as desired/necessary. While this is important, it is unclear from the survey and the interviews just how many (if any) of the respondents had quantifiable, rigid needs for expert systems which could generate a response in a guaranteed time frame. Certainly few of the system developers had formally analyzed or documented any "hard" real-time constraints.

Issues Independent of A System Being an Expert System

An important, but difficult, aspect of analyzing expert system development methodology is distinguishing properties of expert systems that are significantly different from properties of conventional software [2]. This is also an important aspect of the analysis of this survey of V&V issues. Several comments appeared to be due more to factors other than the fact that the system being developed was an "expert system." The interviews helped clarify this issue, and the important ones are discussed in this section.

Extensive Use of Prototyping and Rapid Development

The conventional waterfall life-cycle model has proven to be ineffective for conventional software development. Therefore, it is no surprise that developers do not want to use it for expert system development. A more iterative model (e.g., the spiral model) that includes the use of rapid prototyping is being perceived as a better alternative to the waterfall model. "Conventional" software development projects often include the use of prototyping for activities like developing better user interfaces and having developers better understand the problem domain. These kind of issues are not unique to expert system development, but did come up often in the survey, particularly during the interviews.

Small/Simple vs. Large/Complex Systems

Although some of the systems surveyed are fairly large (e.g., 200 person-months), they are generally much smaller than dedicated software development projects (e.g., Shuttle mission control center (MCC), Shuttle flight software, etc.). The systems surveyed seem to be isolated efforts to develop off-line applications for niches for which expert system technology was felt to be very suitable. They were generally systems that were not part of a larger software system, though they are often used in conjunction with a large data processing system (e.g., they receive real-time data from a large data processing system). This allowed the expert system developers to work without many of the constraints imposed on larger systems (e.g., tightly controlled configuration management).

Addressing a Knowledge Engineer Instead of a Programmer

Although we did not intend to gather information on the experience and background of individual expert system developers, we did learn that several respondents involved in developing expert systems are experts in a problem domain without significant programming experience. This fact was important when formulating the detailed recommendations (discussed in [9]).

Issue Summary

It may be the case that the above issues are indeed typical of expert system development projects and that they should be addressed when addressing V&V of expert system problems. However, it should be recognized that they are somewhat different than the other issues that are true of all expert systems regardless of their size and who is developing them. This may point to a need to tailor suggestions for V&V of expert systems to considerations such as the size of the expert system, the experience of the developer, whether the system is embedded in a much larger software system, etc.

Recommendations Based on the Survey

The major goal of this survey was to discover and document the current state of the practice in V&V of expert systems. Based on the survey results, it appears that much can be done to improve the practice. As a starting point, recommendations for improving KBS V&V were drawn from the survey and interview results. These recommendations are separated into two categories: **direct recommendations** which are directly supported by the survey results and **inferred recommendations** which can be inferred from the survey results by analyzing relationships among the responses.

Direct recommendations include:

- Develop requirements for expert system verification and validation
- Address most often encountered issues
- Recommend a life-cycle for expert systems development

Inferred recommendations include:

- Address readability and modularity issues
- Address configuration management issue
- Develop criteria to classify expert systems by intended use
- Investigate applicability of analysis tools

Survey Conclusions

The original goal of our survey was to gather data and document the current state-of-the-practice in KBS V&V. The survey and follow-up interviews have given us considerable insight into the kinds of problems that developers have really encountered in developing and verifying expert systems. Many of these problems will require additional work before solutions will be readily available. The analysis of the survey and interviews and the subsequent recommendations can serve as valuable reference for directing future KBS V&V research into those areas which are of the most value to KBS developers and users. In addition, managers of KBS development projects can learn from these results to structure life-cycle approaches for KBS development which are more likely to lead to high quality application software.

SPACE STATION FREEDOM PROGRAM V&V REQUIREMENTS ANALYSIS

There are several software V&V requirements for the Space Station Freedom Program (SSFP) that are contained in SSFP documents. KBS V&V issues were not considered when these requirements were defined so it was felt that they might not be appropriate for the V&V of KBSs. To understand the scope of this problem and how it might be resolved, we defined a task to:

- Identify all SSFP V&V requirements
- Analyze the applicability of the requirements to KBSs
- Make recommendations so that all V&V requirements would apply to KBSs. A recommendation could be to change an existing V&V requirement or to develop a KBS V&V technique that could be used to satisfy a requirement.

(A more detailed discussion of this work is discussed in [10].)

Analysis

From several SSFP documents, we initially identified 93 SSFP V&V requirements which were specific to the technical work of software V&V. That is, we did not consider hardware requirements, general documentation requirements, or logistical requirements such as reporting procedures. Grouping similar requirements together and eliminating some minor duplication resulted in 50 distinct requirements.

We analyzed each of the 50 requirements to answer the following questions:

- What is the intent of this requirement ?
- Does this requirement make sense for a KBS ?
- Is this requirement currently satisfied in the current state-of-the-practice ?
- If it is not in the current state-of-the-practice, is there any inherent reason it could not be satisfied ?
- If there is no inherent reason it can not be satisfied, what is it about KBS development that makes this requirement difficult to satisfy ?

Results

Twenty-seven of the requirements are defined either at a level of generality or at a point in the life-cycle where specific software attributes are indistinguishable and can be applied equally to both KB and conventional software systems. Seven of these requirements can be applied to KBSs using existing processes. Thus, 16 requirements remained that were uniquely difficult or impossible to satisfy for KBSs.

We learned that many requirements that would be difficult to satisfy for KBSs were due to two major factors: "life-cycle model" (four requirements) and "system requirements" (five requirements). The "life-cycle model" factor existed because a general waterfall-type of life-cycle model was assumed to be used for system development. For example, the SSFP configuration management requirements would be difficult to apply to an highly iterative life-cycle by having a high overhead to document and release changes to the system. The "system requirements" issue existed because many of the requirements relied on the existence of a detailed set of requirements that identified many considerations; the general state-of-the-practice definitely does not include the generation of such detailed requirements. For example, there is an SSFP requirement to verify quality requirements yet there is no well-understood way of measuring the quality of a KBS.

The remaining V&V requirements that would be a problem for KBSs are:

- Identification of modules (There is no clear way of identifying "chunks" of knowledge as a module, e.g., a rule grouping.)

- Verifying maintainability (It is not clear what makes an expert system maintainable.)
- Requirements to code mapping (Can not be mapped to modules unless modules can be identified; mapping to individual rules/frames is too difficult.)
- Performance analysis (It is difficult to analyze the response time of non-procedural programs.)
- Path coverage (Paths in the conventional sense do not apply to non-procedural programs, paths in a broader sense are much more difficult to identify in non-procedural programs.)
- IV&V (Because of the heavy reliance on experts to aid in verification, independent verification [without the expert or using a different expert] may not be feasible.)
- Verifying off-the-shelf-components (There are not standards in KBS languages as there is in the standard procedural language, Ada.)

Implication to Other Programs

Most existing programs have V&V standards and guidelines that are similar to the SSFP V&V requirements and were generated with conventional procedural software in mind. An analysis similar to the one summarized here would be necessary to adapt the existing program standards and guidelines so they could be applied to KBSs. This approach would be preferable to generating a separate set of standards and guidelines for KBSs. As with SSFP, it is likely that the majority of standards and guidelines could be applied to KBSs without any difficulty so there would not be much duplication. Also, in practice, it may not be clear where in the system a KBS ends and conventional software begins. It may even be the case that a system that starts out being a KBS might end up being implemented as conventional software or visa versa. So having separate KBS and conventional software V&V standards and guidelines would create many difficulties.

SUMMARY

From the survey that we have performed, we have determined that there are some issues with respect to the state-of-the-practice in V&V of KBSs. We have also learned about common practice as well as problems. From the analysis of SSFP V&V requirements, we have learned that conventional V&V standards and guidelines are not completely applicable to V&V of KBSs. We have also learned that the state-of-the practice in conventional software V&V (as represented by standards and guidelines) is significantly different than the state-of-the-practice in KBS V&V.

REFERENCES

1. Bell, M.Z., "Why Expert Systems Fail," JOURNAL OF THE OPERATIONS RESEARCH SOCIETY, Vol. 36, No. 7, 1985, pp. 613-619.
2. Culbert, C., Riley, G., & Savely, R.T., "An Expert System Development Methodology Which Supports Verification and Validation," In PROCEEDINGS OF ISA 88, Instrument Society of America, Houston TX, 1987.
3. Ericson, K.A., & Simon, H.A., PROTOCOL ANALYSIS, MIT Press, Cambridge MA, 1984.
4. O'Keefe, R.M., & Lee, S., "An Integrative Model of Expert System Verification and Validation," EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, Vol. 1, No. 3, 1990, pp. 231-236.
5. Rushby, J., "Quality Measures and Assurance for AI Software," NASA Contractor Report No. 4187, Houston TX, 1988.
6. Slagle, J.R., & Gardiner, D.A., "Knowledge Specification of an Expert System," IEEE EXPERT, Vol. 5, No. 5, 1990, pp. 29-33.
7. Waterman, D.A., A GUIDE TO EXPERT SYSTEMS, Addison-Wesley, Reading, MA, 1986.
8. Constantine, M.M, & Ulvila. J.W., "Testing Knowledge-Based Systems: The State of the Practice and Suggestions for Improvement," EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, Vol. 1, No. 3, 1990, pp. 237-248.
9. Hamilton, D., Kelley, K., & Culbert, C., "State-of-the-Practice in Knowledge-Based System Verification and Validation," To appear in EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, 1991.
10. "Expert System Verification and Validation Study," RICIS Contract #069, Phase 2 - Requirements Identification, Delivery 2 - Current Requirements Applicability, University of Houston / Clear Lake, 1991.

N93-11946

AN APPROACH TO INTEGRATING AND CREATING FLEXIBLE SOFTWARE ENVIRONMENTS

Kirstie L. Bellman, Ph.D.
Computer Science and Technology Subdivision
The Aerospace Corporation
El Segundo, California

Engineers and scientists are attempting to represent, analyze, and reason about increasingly complex systems. Because of the complexity of these systems, no single analysis, model, approach, or viewpoint is sufficient. Such complex systems require not only the availability of a variety of analysis tools, knowledge bases, databases, and programs of all sorts, but also a framework within which these different programs, types of information, and viewpoints can be brought together. Software developers have responded to these needs by introducing the concept of a software "environment." In an environment, the user has access not only to a large number of different "tools" (e.g. analyses, editors, other programs), models, and databases, but often a number of "utilities" and features in the environment that make it easier to go from one tool or model to another. Often these environments have a diversity of knowledge representations (procedural code, equations, text, rules) and languages. Many environments are extendable in at least a limited manner to the languages and information styles already available in the system. However, new languages and representations are being developed continuously for very good reasons: as with mathematical formalisms, a good language can make certain problems easy to do.

Many researchers have been developing new ways of creating increasingly open environments (See Purtilo et al., 1985; Erman et al., 1986; Bond and Gasser, 1988 for examples). In our research on *VEHICLES*, a conceptual design environment for space systems, we have been developing an approach (called ***wrapping***) to flexibility and integration based on the collection and then processing of explicit qualitative descriptions of all the software resources in the environment (Bellman and Gillam, 1990; Landauer, 1990). The detailed descriptions (or metaknowledge) of the resources are used by the system to help partially automate the combination, selection, and adaptation of tools and models to the particular requirements of the user and the type of problem being solved. This approach also allows for a great diversity of information types and languages. At the current time, we have a simulation, *VSIM*, used to study both the types of wrapping descriptions and the processes necessary to use the metaknowledge to combine, select, adapt, and explain some of the software resources used in *VEHICLES*. Below, we briefly describe what we have learned about the types of knowledge necessary for our wrapping approach and the implications of wrapping for several key software engineering issues.

The *VEHICLES* environment is composed of both conventional and artificial intelligence methods and programs. It is a distributed, multilingual environment that is largely written in Prolog, C and C++, but also supports external programs written in a diversity of languages. It supports a variety of information and knowledge types, multiple models, and a broad toolchest of analyses, graphics, and other types of software programs. Although it is a prototype environment, in its four years of development it has been used to provide some of the analyses supporting several space programs. As noted above, supporting the design and analysis of complex systems requires a diversity of models and tools; the result is often a software environment that becomes itself a complex system. Hence, we feel it is important to provide ***intelligent user support functions***; that is some means of supporting the

user (be it human or another computer program) in the selection, assemblage, integration, adaptation, and explanation of the software resources.

By ***selection***, we mean that the system helps the user to select which software resources are appropriate given the current problem or task. For example, in *VSIM*, we have experimented with two simple scenarios involving selection: in the first case, a human user has selected *optimize* from a menu containing a number of analyses in *VEHICLES* and the system uses the wrappings of three optimization programs and the wrapping for the set of equations to be optimized to determine which optimization program is most appropriate; when the system finds no basis for distinguishing between two of the optimization programs, it uses wrappings again to select an appropriate user screen for presenting the user with the remaining candidate optimization programs from which to select. In the second case, a *VEHICLES* solver has bombed on a set of equations and the system itself poses the problem of selecting another solver, which is done automatically on the basis of the wrappings, with a record kept of the choice and use of the selected solvers.

To us ***integration*** is more than simply allowing tools to 'talk' (we prefer to use the term ***assemblage*** for this permissive hooking-together of tools); rather, it is providing some means for deciding when tools should talk. For example, when should a given model send its output to another model; when should a given database provide the information for a given analysis. In the wrappings, we have conditionals (implemented as rules, but there could be other implementations) which help define the context for integrating tools and models.

By ***adaptation***, we mean the modification of the software resource depending upon the problem or task and the information currently available. This adaptation could be changing the input file or control parameters to a simulation or changing the queries to a database or changing the default values in a model and so forth. The last critical intelligent user support function is ***explanation***, that is, at a minimum, providing the means to record and document how the software resources were selected, integrated, and modified during the use of the software environment. Eventually, we would like a more interesting form of explanation, where the explanation is adjusted depending upon the user and the problem or task.

Using *VSIM*, we have learned a number of things about the knowledge necessary in wrapping, which we summarize below. First, in order to perform the five intelligent user support functions listed above, we need to represent and utilize three types of knowledge: metaknowledge (e.g. knowledge about a given method or tool or about the use of knowledge in a knowledge base), user models (knowledge about the types and activities of the user), and domain knowledge (especially knowledge about the types of problems in that domain and the types of contexts that constrain the choice and use of given methods and information.) In *VSIM*, we have been experimenting on how to utilize each type of knowledge; currently *VSIM* is composed of a *planner knowledge base (PKB)*, a *wrapping database (WDB)*, and a set of wrapping processors and other software resources, which are all wrapped. The PKB contains triplets of the form:

{Problem Definition
Information Available
Resource Name}

The "problem definition" has been simplified to be a list of keywords corresponding to the activities that the system can provide to the user, such as "optimize", "solve", "parametric study"; or at a higher level, they could be such activities as "design a new satellite" or "tailor an existing satellite". Eventually, we can incorporate more interesting problem decomposition methods; we have simplified the problem definition in order to study how to relate the problem descriptions to the software resources, and how to specify the minimal information required by the software resource to be used for a given problem. In the WDB, each wrapping contains a name of a software resource, input and output requirements and restrictions, and then we have been experimenting with many different ways of expressing additional information about the appropriate use of the resource under different conditions. One important point to note is that in *VSIM* all the software resources are wrapped, including all programs processing the wrappings. Hence, *VSIM* selects the "matcher" program used

to match the wrappings of the model and the optimization programs, in the example scenario described above.

In the PKB, the problem definition reflects knowledge about: the resources provided by the software environment (metaknowledge); the desired activities of the user (user models); and the methods and requirements of solving problems in a given domain. In the WDB, the knowledge is largely metaknowledge about the use and type of software resource, but it crosses any neat lines and includes in any conditionals, references to domain knowledge and user models.

One of the problems we encountered when we started to write the wrappings was what we call the "library problem." That is, we tried to formulate a description of a software resource that would be suitable for all wrapping purposes for all time. We soon learned that, at least for the purposes of formulating these descriptions, we need to start with five different descriptions, each containing the semantics corresponding to the five different intelligent user support functions described above. In addition, for a large software resource (such as the large simulations we deal with in *VEHICLES*), we need to develop several wrappings, each corresponding to a major mode of use for that resource. Lastly, an issue we have not yet addressed in *VSIM*, we can not consider the wrappings as a static description. Rather, we need to devise wrapping processes such that the descriptions continue to build, as the resources are used. Similarly, a human user must be able to browse, edit, and add to the descriptions.

Although we have focussed on the flexibility and integration provided by utilizing wrappings, it is important to emphasize that flexibility and integration in a software environment occur at several different levels. Hence, in addition to the use of wrappings, we have also experimented with how best to use network services and message-passing kernels to take advantage of different programming languages and platforms.

The wrapping approach also advances software engineering in several significant ways: 1) it provides explicit descriptions (and documentation) about each software resource, including what is in essence both a specification for that resource and practical advice on its acceptable and appropriate use; 2) it provides traceability during dynamic testing, and an easy way to insert probes; 3) it allows standard structural testing of the wrappings, when these are stored as a knowledge base/database; 4) it allows the possibility of incorporating on-line software checkers.

The wrappings are descriptions in a database/knowledge base. Hence, a number of standard static testing and analysis strategies that have been applied to knowledge bases (see Landauer, 1990; Bellman, 1990), can be applied to the wrapping database. For example, static analyses can check to see if a given resource is used by any other resource; standard type checking can pick up not only lower-level information about data types, but also new higher-level information about the type of resource and uses that were made explicit in the wrappings. With a simulation such as *VSIM*, one can dynamically test the interactions among different software resources. Wrappings can provide an interesting means of seeding errors, adding special programs to provide intermediary values or other types of debugging information, or altering the combination of resources. When combined with 'user log'(in *VSIM*) and self-documentation (in *VEHICLES*) programs, this approach offers the ability to perform and record a large variety of software engineering experiments.

Lastly, the wrappings represent a self-description of a software environment that is processible by that environment. In Maes' terminology (1987), such a system is "computationally reflective" and her everyday examples of reflection range from the now commonplace, e.g. keeping performance statistics and debugging information to the exciting possibilities for autonomous systems and programs with self-optimization, self-modification, and self-activation. We are excited by the recent realization that *VSIM* can eventually be considered just another resource in the *VEHICLES* environment; one with the rather special property of being a simulation of itself. Hence when we add a new resource to *VEHICLES*, we would eventually be able to immediately simulate its integration into the system. With wrappings, we hope to make software architectures more testable, maintainable, and open. The hope is that eventually we will have computer systems in which the means to test and evaluate the system are not peripheral, but rather an integral part of the

software system.

References

Bellman, Kirstie L. The Modeling Issues Inherent in Testing and Evaluating Knowledge-Based Systems. Expert Systems With Applications, Vol. 1, 1990. (Pergamon Press)

Bellman, Kirstie L. and A. Gillam. Achieving Openess and Flexibility in Vehicles. In AI and SIMULATION Theory and Applications. Proceedings of the SCS Eastern Multiconference, 23-26 April, 1990, Nashville, Tennessee. Simulation Series Vol. 22(3), April 1990. pp 255 -260.

Bond, A.H. and L. Gasser, editors. Readings in Distributed Artificial Intelligence. Los Altos, Ca: Morgan Kaufmann, 1988.

Erman, Lee D., Jay S. Lark, Frederick Hayes-Roth. Engineering Intelligent Systems: Progress Report on ABE. Teknowledge Inc TTR-ISE-86-102. In Proceedings: Expert System Workshop, April 1986. SAIC Report Number SAIC-86/1701.

Landauer, Christopher. Correctness Principles for Rule-Based Expert Systems. Expert Systems With Applications, Vol. 1, 1990. (Pergamon Press)

Landauer, Christopher. Wrapping Mathematical Tools. In AI and SIMULATION Theory and Applications. Proceedings of the SCS Eastern Multiconference, 23-26 April, 1990, Nashville, Tennessee. Simulation Series Vol. 22(3), April 1990. pp 261 -266.

Maes, Pattie. Concepts and Experiments in Computational Reflection. OOPSLA '87 Proceedings, 1987. pp 147 - 155.

Purtilo, James. "POLYLITH: An Environment to Support Management of Tool Interfaces", ACM 0-89791-165-2/85/006/0012. 1985.

Purtilo, James M. "POLYLITH and Environments for Mathematical Computation", University of Illinois Dept of Computer Science, Report No. UIUCDCS-R-84-1135. 1984.

N93-11947

Author: Captain F Jesse Fanning
Date: 11-Dec-1990
Posted-date: 21-Feb-1991
Subject: abstract '90

Design of an Ada Expert System Shell for the VHSIC Avionic Modular Flight Processor

Abstract

The Embedded Computer System Expert System Shell (ES_Shell) is an Ada-based expert system shell developed at the Avionics Laboratory for use on the VHSIC Avionic Modular Processor (VAMP) running under the Ada Avionics Real-Time Software (AARTS) Operating System. The ES_Shell provides the interface between the expert system and the avionics environment, and controls execution of the expert system. Testing of the ES_Shell in the Avionics Laboratory's Integrated Test Bed (ITB) has demonstrated its ability to control a non-deterministic software application executing on the VAMPs which can control the ITB's real-time closed-loop aircraft simulation. The results of these tests and the conclusions reached in the design and development of the ES_Shell have played an important role in the formulation of the requirements for a production-quality expert system inference engine, an ingredient necessary for the successful use of expert systems on the VAMP embedded avionic flight processor.

Session I7: PLANNING AND SCHEDULING

Session Chair: Chris Culbert

AUTONOMOUS POWER SYSTEM: INTEGRATED SCHEDULING

Mark J. Ringer
Sverdrup Technology Inc.
NASA Lewis Research Center Group
Brook Park, Ohio 44142

ABSTRACT

The Autonomous Power System (APS) project at NASA Lewis Research Center is designed to demonstrate the abilities of integrated intelligent diagnosis, control and scheduling techniques to space power distribution hardware. The project consists of three elements: the Autonomous Power Expert System (APEX) for fault diagnosis, isolation, and recovery (FDIR), the Autonomous Intelligent Power Scheduler (AIPS) to determine system configuration, and power hardware (Brassboard) to simulate a space-based power system. Faults can be introduced into the Brassboard and in turn, be diagnosed and corrected by APEX and AIPS.

The Autonomous Intelligent Power Scheduler controls the execution of loads attached to the Brassboard. Each load must be executed in a manner that efficiently utilizes available power and satisfies all load, resource, and temporal constraints. In the case of a fault situation on the Brassboard, AIPS dynamically modifies the existing schedule in order to resume efficient operating conditions.

A database is kept of the power demand, temporal modifiers, priority of each load, and the power level of each source. AIPS uses a set of heuristic rules to assign start times and resources to each load based on load and resource constraints. A simple improvement engine based upon these heuristics is also available to improve the schedule efficiency.

This paper describes the operation of the Autonomous Intelligent Power Scheduler as a single entity, as well as its integration with APEX and the Brassboard. Future plans are discussed for the growth of the Autonomous Intelligent Power Scheduler.

1. INTRODUCTION

Many of the obstacles in the implementation of an on-board integrated scheduling and FDIR system with actual hardware have yet to be investigated. The APS project is meant to research these problems with a set of increasingly complex software and hardware systems. Details of the scheduling aspect of the APS project will be discussed in this paper. A brief graphic of the APS system, each unit's functionality, and communicated information can be seen in Figure 1.

1.1 The Need For Scheduling

Aboard a complex space-based system, many activities must be performed, each competing for a multitude of temporal positions and resources. A scheduler must assign start times to each activity without violating any resource or temporal constraints. Many of the resources aboard such a spacecraft will be vastly oversubscribed, having many times more resource requests than available resources. This makes it a paramount objective to efficiently utilize the available resources in order to complete as many activities as possible.

An automated scheduler can be included within a spacecraft, schedule data can be transmitted from a ground base to the spacecraft, or a combination of the two can be employed to accomplish the task of spacecraft scheduling. On-board automated schedulers are still in the research stage and are the topic of this paper. Current Space Station Freedom designs dictate that the schedule be generated on the ground and sent up to the Space Station. All scheduling and rescheduling will be done on the ground [Hagopian 1990]. It may also be possible to have two schedulers, one on the ground, and one in space. The ground-based scheduler will be able

generate very efficient schedules with detailed domain knowledge and large host computers. The on-board scheduler will be able to work on the same problem in "real-time", but do so less effectively because of time and processing constraints.

The problem of scheduling is complex in the areas of computation, domain knowledge, and quantity of data. Computationally, scheduling can be classified as an NP- Hard problem. It is very easy to give a common problem, such as scheduling one day of Space Station experiments, that cannot be implicitly solved in a time comparable to the lifetime of the universe, even on a supercomputer! Building a scheduling engine with enough domain knowledge to be capable of complex reasoning in the area of scheduling is a difficult task. It is also a large bookkeeping task maintaining sufficient information on activity and resource attributes.

1.2 The Need For Automated Scheduling

Future large spacecraft will require larger and more sophisticated infrastructure systems and living environments. Such spacecraft will consist of dozens of resources and hundreds of attached loads. The electrical power system on a platform such as the Space Station Freedom, Lunar base, or Mars base represents a critical portion of such a system. The APS project explores intelligent hardware and software architectures for efficient system operation and scheduling on a representative electrical power system [Ringer 1990, Ringer 1991].

Standard scheduling concerns are beyond the capabilities of humans onboard the spacecraft as well as counter-productive to the science activities planned for the crew of such a vehicle. Ground-based systems are another possibility for generating and implementing schedules. The scheduling computers and humans on the ground would be responsible for schedule generation, while on-board non-scheduling processors would be responsible for the minute-by-minute implementation of the schedule. If the scheduling expertise and computers are kept on the ground, every anomaly that occurs aboard the spacecraft that incurs a schedule change would cause significant time delays and efficiency losses. The relevant information will have to be sent from the spacecraft to the ground, a solution generated, and the information transmitted back to the spacecraft. This will cause a lengthy problem solution time caused by communication and processing delays. This communication delay is compounded for systems such as a Mars base or deep space probes [Dolce 1990].

Since the control of such a large system is difficult to accomplish with the use of ground-based systems, the importance of an automated on-board system is evident. Automated scheduling here means not only automated schedule generation, but also automated schedule implementation. The more responsibilities that an automated system assumes will increase the speed of critical decisions. The use of on-board scheduling for control of such a system will significantly decrease operating costs, increase efficiency, and provide for safer operation [Ringer 1991].

2. AIPS IMPLEMENTATION

The AIPS scheduling problem can easily be broken down into three main areas: schedule representation, schedule generation, and interaction with the other subsystems of APS. Schedule representation is the internal description of the schedule as well as description of the activities and resources modelled in AIPS. Schedule generation describes the scheduling engines used to build the schedules. Interaction describes the interface to APEX and the Brassboard as well as the graphical user interface. Figure 2 shows the main AIPS interface and will be referred to in the next three sections. The remainder of the paper will give an in-depth discussion of each of these three areas.

2.1 Representation

APEX keeps system configuration information, basically representing which loads (activities) are attached to which power sources. Resource availability and activity descriptions are sent to AIPS, which assigns start times to each activity.

The usual method for scheduling continuous-time problems is to break the problem down into multiple subproblems, or planning horizons. These planning horizons represent customary units of time, like one day. Problems of this size are easier to tackle than scheduling all activities for the lifetime of the spacecraft. Once the problem is broken down though, it is necessary to put the pieces back together to create a time-coherent sequence of events. In the AIPS scheduler, it is possible for activities to cross over different planning horizons. It would be unreasonable to assume that at the planning horizon break, all activities would also have a natural break point.

2.1.1 Resources

Capacity type resources are the only type modelled since electric power is the only resource currently available on the Brassboard. Each activity

can only be connected to one power source at a time. There is no representation of power sharing among different power buses, but this case can be represented by combining two sources into one. This connection information is forced by the Brassboard configuration.

Each resource consists of a time-varying profile of available power for the current planning horizon. It is also necessary to include some information about the power availability in the next planning horizon, as discussed in the previous paragraph.

2.1.2 Activities

Each activity is an entity that requests a certain resource (power), from the power sources. (Note: in the scheduling domain this entity is called an "activity", while in the Brassboard domain it is referred to as a "load"). Each activity consists of certain attributes including duration and a list of time variant resource requests. An activity's priority is a relative measure of the activities importance to the user of the activity completing. The activity must also specify to which resource that it is attached. This is actually the job of APEX, which keeps system configuration information. Based upon switch positions within the Brassboard, different paths can be implemented between sources and loads.

Each activity can also have temporal modifiers which specify a time window in which the activity needs to finish. It can also request a general position preference on the timeline, be it early, late, or in the middle of the scheduling horizon or specified time window.

2.2 Generation

Schedule generation is the process of assigning resources and temporal positions to activities. Many types of scheduling engines can be used, with each one having various strengths and weaknesses. Two main factors that can be used to judge a scheduling engine are time to generate a schedule and the schedule's "goodness". These two factors can be compared if the both engines represent the same problem fully.

Some may argue that scheduling and rescheduling are the same problem, with scheduling represented as rescheduling with no activities currently on the timeline. This may be true, but it may still be advantageous to separate the two. In most scheduling domains, a "reasonable" amount of time exists to generate a schedule. In most rescheduling domains, "much less" time is available, any time spent rescheduling the problem is wasted time, i.e. a loss in resource-usage efficiency. For this reason, different scheduling engines are used for the two cases. There is, of course, a tradeoff between the two scheduling methods, faster schedule generation time is traded for less schedule "goodness", but this is made up in the reclamation of the lost time between the anomaly and the "slow" schedule re-implementation (rescheduling).

The rescheduling engine is a heuristic non-backtracking (to re-visit either unscheduled activities, or unchecked time periods) engine that produces a reasonably efficient schedule in a short time. The improvement engine is based on the previously stated engine, but is able to improve the schedule iteratively. Other scheduling engines implement the same concept by using more or less knowledge and/or scheduling methods in the decision-making processes during schedule generation depending on the amount of time allowed [Britt 1990]. Other work in multiple scheduling engines has been focused in constraint-based simulated annealing repair techniques [Zweben 1990a].

The AIPS scheduler and rescheduler use a set of schedule building heuristics based on global knowledge of the resource-based scheduling problem. In this way, a reasonably efficient schedule can be created. The same set of heuristics can also be used in the case that activities need to be removed from the timeline or repositioned on the timeline. The heuristics used in the AIPS scheduling engine can be divided into two categories: those used to select which activity to schedule and those to determine what temporal placement to assign each activity [Sadeh 1989].

2.2.1 Activity selection heuristics

Since the scheduler does not backtrack it is necessary to determine in which order to place activities on the schedule. An ordering of activities is created based on predicted importance to place on the schedule. Once this list is compiled, each activity is then placed on the schedule, if it does not fit, it is ignored and not scheduled. In a non-backtracking engine, an activity is more likely to be scheduled and scheduled in its desired position if it is placed on the schedule as early as possible in the scheduling process. It should be stated that each of the next three classes of heuristics are not used on their own, but a combination of the three is used to generate the ordered list of activities.

> Priority: It is more important to include the higher priority activities on the schedule, therefore, higher priority activities are placed on the schedule first.

Amount of Power Requested: Placing larger resource amount requesting activities on the schedule first usually result in schedules that will consume more of the capacity type resources. If one places smaller resource requests on the schedule first, it is possible that the larger activities will be unable to fit on the schedule. This would result in less capacity type resource usage.

Duration Of Requested Time Window: If an activity has requested that it be placed in a small time window, it is important that this activity be placed on the schedule before others that may take resources within this time window needed by the current activity.

2.2.2 Temporal Placement Heuristics

It should again be stated that the next three classes of heuristics do not stand alone, but all three are used to determine the final position of an activity on the timeline. Each feasible start time for an activity is judged based on the following heuristics. The best of these possible start times is then chosen, and the activity is placed on the schedule.

Projected Resource Demand: This is a conflict avoidance strategy, as opposed to a reactive conflict resolution strategy [McLean 1990]. Bottleneck areas (areas of resource oversubscription) are projected before the schedule is generated therefore they are avoided during schedule generation. Resource bottleneck projection is affected by specified time windows as well as temporal loading preferences. A projected demand for each resource is calculated before any actual scheduling is done. This bottleneck information is used to affect the placement of each activity on the schedule. As each activity is placed on the timeline, the bottleneck areas are updated. This strategy allows the scheduler to move some activities away from bottleneck regions, thus allowing more activities to be scheduled that would have been unscheduled because of resource conflicts.

Closeness to loading preference: This is a relative measure of how close an activity is to it's requested position on the timeline. If an activity has requested front loading, it would be preferrable to place the activity at the beginning of the timeline as opposed to the end.

Front Loading: In the representation of the schedule, it is possible to have an activity continue into the next planning horizon. This is possible but not desirable, since less of the activity is being accomplished in the current horizon. In real life terms, one can always do today's work tomorrow, but then tomorrow's work will be delayed until the next day and so on. For this reason, it is more desirable to complete the activity within the present planning horizon.

2.2.3 Improvement Engine

A very simple iterative improver has been constructed based on the heuristic rules explained above. The word "improver" is used here to signify a better schedule, as opposed to "optimize" which implies an optimum schedule. Since the scheduler's heuristics are using incomplete knowledge of the problem, it is not "true" that the schedulers' decisions are always "optimal". In other words, even though the heuristics produce a reasonable schedule, they are by no means producing the optimum schedule. For this reason, some noise added into the decision making processes can actually produce a better overall schedule. The schedule generation process can be repeated, each time the schedule is generated, a bit of randomness is added to the decision making process. By keeping the last best schedule in memory, the scheduler can "search" for a better schedule, until the time that the schedule must be implemented.

This is somewhat similar (but opposite) to the simulated annealing approach to scheduling. In an annealer, the actual activity placement makes for a currently worse schedule but in the long run, a better schedule is produced. In the AIPS heuristics, the decision knowledge is incomplete, therefore, a worse heuristic decision may actually produce a better overall schedule when the entire scheduling process is complete.

The first time the heuristics produce a schedule, a feasible schedule is generated in a short period of time. Since the last best schedule is saved, this engine can still be considered an "anytime" scheduling engine, because at anytime in the scheduling process (after the first schedule is generated), a feasible schedule exists [Zweben 1990b]. As a counter-example, some schedulers may use so much knowledge in constructing a schedule, that they make take a significant amount of time to construct a schedule and can not be stopped in the middle of the process if a schedule is needed.

2.2.4 Non-nervous rescheduling

The same set of heuristics are also used in the process of rescheduling. The idea of non-nervous (small perturbation to the existing schedule) rescheduling is meant to save time in the outside world as well as within the spacecraft. Many other

activities, and maybe even humans on the ground may be waiting for the execution of a certain activity. If, during rescheduling, the scheduler makes large changes to the current schedule, many activities or even humans will be affected by the changes.

In some cases, the AIPS rescheduler only finds it necessary to shed a certain number of loads. This shedding is the effect of source reductions and is heavily influenced by activity priority. In other cases, certain activities may need to be deleted from the schedule, and other activities may be available to take the resources abandoned by the deleted activity.

2.2.5 Judgement of "Goodness"

Schedule judgement is needed in the iterative improver, for comparing two schedules to see which one is better, or for the user as a relative measure of "goodness". Judging a schedule is a not an exact science with many ways of looking at the problem. Of course, in a software environment, more concrete measures of schedule "goodness" are necessary. Many possibilities of judging a schedule exist, the AIPS schedule judge uses three main attributes of a completed schedule to serve as a judgment of a "good" schedule. The three attributes are power use, priority weighted number of activities scheduled, and activity position.

Power use is the amount of available power consumed by the scheduled activities. This is kept as a ratio of power used to power available. The number of loads scheduled factor is weighted by the priority of each load scheduled. It is possible to have activities continue into the next planning horizon, if this happens, the number of loads scheduled factor is penalized because the activity is not being completed in the current planning horizon. The load position factor is a measure of how close each load is to it's preferred temporal loading position.

The bottom portion of Figure 2 shows some of the schedule judgment criteria (all normalized from 0 to 1). "Time", "Counter", and "Best" have no meaning in this example. "Current" represents an overall schedule goodness rating, "Energy Used" represents the ratio of energy used in the current schedule to the energy available, "WNOL scheduled" stands for Weighted Number Of Loads scheduled, "Closeness" is a measure of how close each activity is to it's specified loading preference, and "Demand" represents the ratio of energy requested to energy available.

2.3 Interaction

There are two types of interaction; between APEX and AIPS, and between AIPS and the user. The interface between the user and AIPS is meant for testing of the scheduler, displaying information to the user, and human interaction for semi-autonomous control of the scheduler. The interface to APEX is used to implement the schedule that is generated, and to transmit rescheduling information to APEX.

2.3.1 Graphical Interface

The graphical user interface can be broken down into three major sub-views: resource, timeline, and activity displays. The resource displays are shown at the top of the screen as line graphs. The timeline is shown in the middle of the screen as a Gantt chart. The activity is shown at the bottom of the screen representing resource requests and activity information.

The scheduler has an interactive graphical user interface that can be used to test the scheduler as shown in Figure 3. This interface is fully mouse controlled and allows the user to edit activity information, resource information, as well as making changes to the schedule after the scheduling engine has given its solution. The user may test to see if they can manipulate the schedule in such a way to build a "better" schedule than the scheduling engine itself. This user interface also shows more clearly how the scheduling engine functions and makes it easier to test the scheduler. Figure 3 shows a schedule consisting of 30 activities, 2 resources, and is calculated with a time granularity of five minutes.

The upper two graphs show two sources of electrical power, with the power on the y-axis and time on the x-axis. Available power is shown as a dotted line, while scheduled power is shown as a solid line. This difference between available power and scheduled power is the unused (unscheduled) power which is shown as the dotted fill area between the available and scheduled power.

The Gantt chart in the middle of the screen shows each activity that was scheduled as a solid line or a dotted line if unscheduled. The length of the line corresponds to the length of the activity and the scale is shown on the x-axis of the Gantt chart. The earliest start and latest end points (if they are specified) are shown by brackets at each side of the activity. On the color screen, each activity is color coded by its priority. If an activity continues into the next planning horizon, this is shown as an arrow at the end of the activity.

The edit window at the bottom of the screen shows a more specific description of the activity that the mouse is currently pointing to. Values such as power demand, length, start and end constraints, priority, source, and loading preference can be specified within this window. This information is available for each activity on the schedule. In Figure 3 the mouse is pointing to the activity entitled "health", and the information for this activity is shown in the edit window.

2.3.2 APEX Interface

In order to adequately model the interaction between APEX and AIPS, a set of protocols was developed to communicate different scheduling and rescheduling procedures. Protocols were developed to generate an initial schedule and modify existing schedules. The initial schedule generation takes a set of resources and activities and generates a schedule for APEX to follow.

Five modification protocols exist: activity change, resource change, activity add, activity delete, and resource delete. During the execution of a schedule, the priority of an activity may change, the power demand of an activity may change, the activity may need to be dropped from the schedule, or an activity may need to be added. This change information is caused by faults in the Brassboard. Resources also may be changed during the execution of a schedule, or deleted altogether. These protocols enable APEX and AIPS to communicate system configuration information as well as reconfigure the system in the case of a fault.

APEX holds a database of activity information, as well as information on resource availability. This would probably be the job of another higher level computer in a real application, but was a suitable place for the APS project. APEX sends this activity and resource information to AIPS, and AIPS in turn generates a schedule, assigning activities start times, and subscribes to the available power.

3. FUTURE WORK

Work is currently underway in many aspects of the AIPS scheduler. The underlying representation of available resources is being changed from simple arrays to a list of start time, end time, value objects. Representation of consumable resources is being developed with a special emphasis on batteries. Work will continue on improving the heuristics used to generate the schedule and some type of reactive scheduling engine will be developed. Dynamic activity priorities and uncertainty in an activity's resource requirements will be included.

4. CONCLUSION

The AIPS scheduler has shown the ability to function in a dynamic real-world environment. Functions such as schedule generation within a time-limited environment, dynamic rescheduling, and integration with FDIR and hardware systems have been implemented in the AIPS software. Scheduler interaction with other intelligent agents has been demonstrated. These accomplishments along with the functionality of APEX and the Brassboard display automation strategies applicable to larger and more complex systems. Work will continue on APS to prove the feasibility of on-board automation with increasingly complex software systems and demonstrations.

ACKNOWLEDGEMENTS

Thanks to Todd Quinn, Tony Merolla, Walt Krawczonek, and Gene Lieberman for implementing the other portions of the APS project. Thanks to Jim Kish for his work in managing and coordinating the APS project. This work was performed under contract NAS3-25266 with Jim Kish as Coordinator.

REFERENCES

[Biefeld 1990] Biefeld, E., Cooper, L., "Operation Mission Planner: Final Report", JPL Publication 90-16, March 15, 1990.

[Britt 1990] Britt, D.L., Geoffroy, A.L., Gohring, J.R., "Managing Temporal Relations", Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence, 1990, NASA CP 3068.

[Dolce 1990]Dolce, J.L., Kish, J.A., and Mellor, P.A. "Automated Electric Power Management and Control for Space Station Freedom", In Proceedings 25th IECEC, AIChE 1990.

[Hagopian 1990] Hagopian, J., "Short Term Plan Contents", Space Station User Operations Working Group, Mission Planning Workshop, NASA Marshall Space Flight Center, May 1990.

[McClean 1990] McLean, D.R., et al., "Emphasizing Conflict Resolution Versus Conflict Avoidance During Schedule Generation", World Congress on Expert Systems, Orlando, Florida, Dec., 1991.

[Ringer 1990] Ringer, M.J., and Quinn, T.M., "Autonomous Power Expert System", In Proceedings 25th Intersociety Energy Conversion Engineering Conference IECEC, AIChE 1990.

[Ringer 1991] Ringer, M.J., Quinn, T.M., and Merolla, T., "Autonomous Power System: Intelligent Diagnosis and Control", Proceedings of the NASA Goddard Conference on Space Applications of Artificial Intelligence, 1991.

[Sadeh 1989] Sadeh, N., and Fox, M.S., "Focus of Attention in an Activity-Based Scheduler", Proceedings of the NASA Conference on Space Telerobotics, Pasadena, California, 1989.

[Zweben 1990a] Zweben, M., "A Framework for Iterative Improvement Search Algorithms Suited for Constraint Satisfaction Problems", NASA Ames Artificial Intelligence Research Branch Technical Report, February, 1990.

[Zweben 1990b] Zweben, M., Deale, M., and Eskey, M., "Anytime Rescheduling", NASA Ames Artificial Intelligence Research Branch Technical Report, February, 1990.

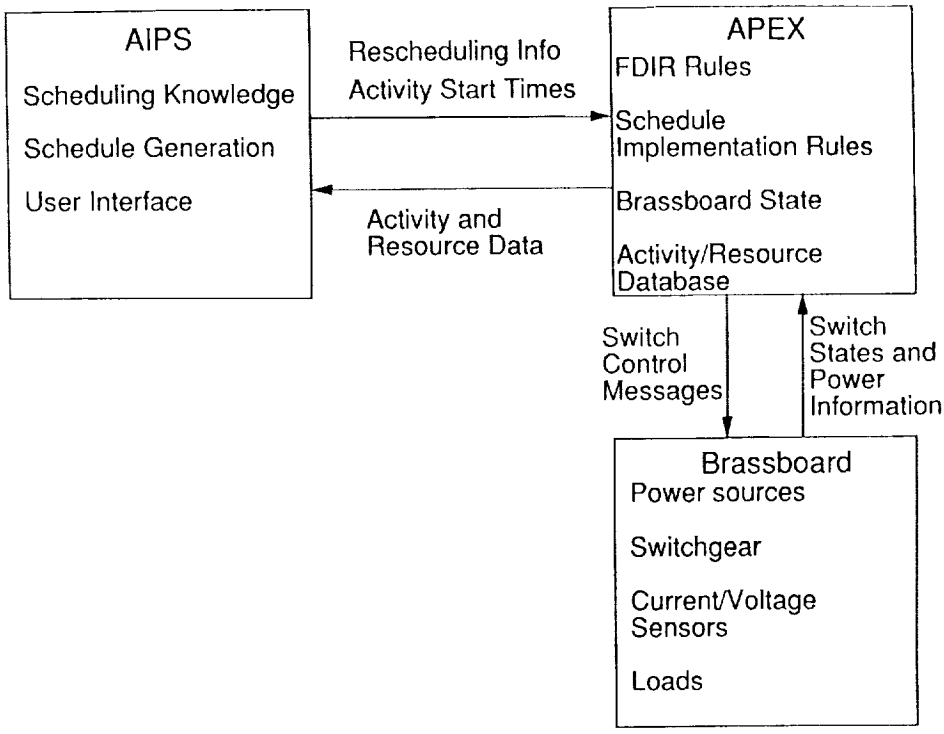


Figure 1. APS Component Functionality and Description

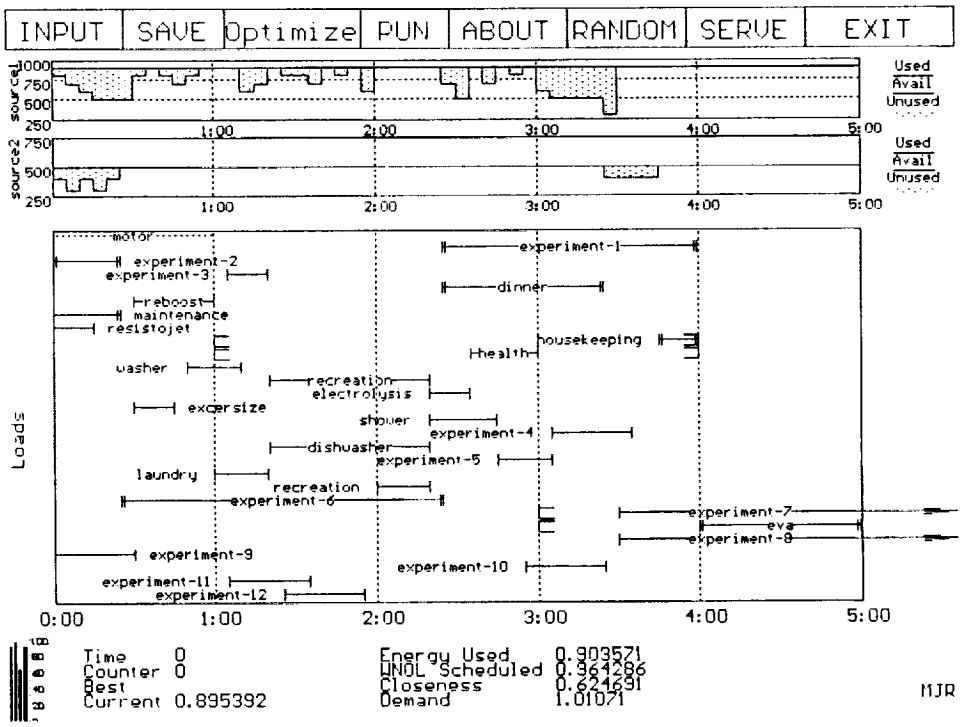


Figure 2. Generated Schedule With "Goodness" Information.

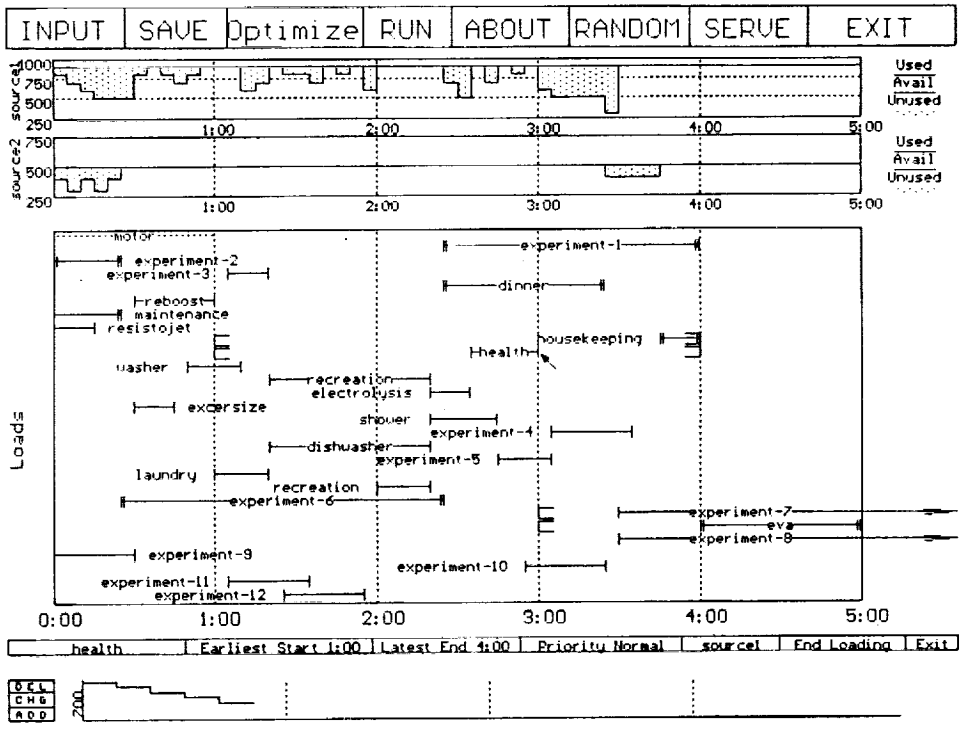


Figure 3. Generated Schedule With Activity Information.

Methodologies for Building Robust Schedules

John H. Dean
McDonnell Douglas Space Systems Company
16055 Space Center Boulevard
Houston, TX 77062
(713) 283-4008

Abstract: COMPASS is the name of a Computer Aided Scheduling System designed and built by McDonnell Douglas Space Systems Company for NASA. COMPASS can be used to develop schedule of activities based upon the temporal relationships of the activities and their resource requirements. COMPASS uses this information, and guided by the user, develops precise start and stop times for the activities. In actual practice however, it is impossible to know with complete certainty what the actual durations of the scheduled activities will really be. The best that one can hope for is knowledge of the probability distribution for the durations. This paper investigates methodologies for using a scheduling tool like COMPASS that is based upon definite values for the resource requirements, while building schedules that remain valid in the face of schedule execution perturbations. Representations for the schedules developed by these methodologies are presented, along with a discussion of the algorithm that could be used by a computer onboard a spacecraft to efficiently monitor and execute these schedules.

Introduction

The dictionary definition of robust is "strong and healthy." A robust schedule, therefore would be one which exhibits the characteristics we associate with strength and health. There are two interesting characteristics of schedule strength. The first is the ability of the schedule to accomplish useful work (how much is scheduled), and the second is the ability of the schedule to resist failure due to perturbations (the reliability of the schedule). Obviously these two characteristics are in competition with each other. A densely packed schedule will be more prone to failure if activities run long when actually executed. Alternatively, padding the scheduled durations of the activities with some extra "slack" time, in order to absorb any perturbations, reduces the number of activities that can fit into a fixed length schedule.

In order to examine the concept of robust schedules, we defined metrics that capture these two differing characteristics of schedule strength. Many metrics for measuring the amount of work that a schedule accomplish have been proposed before. In fact, it is these kinds of metrics that most schedule optimizers use as their objective function to maximize (or minimize). Examples of these kind of metrics include total make span time, summing the values of the activities placed on the schedule, mean or total tardiness, and mean time in process. This paper defines a schedule robustness metric that is a measure of the reliability of the schedule

Metrics for describing the reliability of hardware items is typically described as a Mean Time To Failure (MTTF). Furthermore, models exist which describe the expected reliability of systems built of component pieces for which the stochastic behavior is known, or can be derived. Similarly, our approach develops a notion of a MTTF for a schedule. To do this, we defined a concept of the failure of a schedule, and developed a model that describes how to calculate the MTTF of a schedule, given a description of the stochastic behavior of the activities that make up the schedule.

In order to define the concept of a schedule failure it is necessary to describe the overall schedule development and execution process. Schedule development begins with a set of tasks to be performed, along with there resource requirements. In addition, there may be some temporal relationships between tasks. For example, one task may require that a second task be completed before the first

task can begin. Based upon this information, along with other information such as the task priority or value, a schedule is created. Typically, the schedule that is created will be feasible. In other words, the schedule will contain no inconsistencies. All the given temporal constraints will be satisfied, and none of the resources will be oversubscribed. At execution time, the schedule is used to determine what resources should be assigned to what tasks, and when. As the schedule is executed, deviations from the a priori schedule will occur. If these deviations become too large, the schedule will no longer be valid, and a new schedule of the remaining tasks must be created. When this happens, the original schedule has suffered a failure.

Schedule development consists basically of assigning resources and times for the performance of activities in order to meet some deadline. It is well established that the Resource Constrained Scheduling decision problem (RCS) is NP-complete[1], and most scheduling decisions are NP-hard. This means that the length of time to develop a schedule is of exponential order relative to the number of tasks and/or resources. Since RCS is NP-complete, the time to verify a particular encoding of a solution to a RCS problem is of polynomial order relative to the number of tasks and resources, however. This provides the rationale for the definition of a schedule failure. When the perturbations become large enough that a polynomial bound algorithm can no longer accommodate the deviations, a schedule failure occurs, and the NP-hard problem must be solved again.

The remaining question is the representation of the schedule which can be verified in polynomial time. This paper will describe two schedule representations called the time constrained schedule representation and the order constrained schedule representation. These two representations can be merged into a single approach to allow the schedulers to use their choice of method.

Time Constrained Schedule Representation

The standard definition of a RCS problem is a follows: Given a set T of tasks t_i, for $1 \le i \le n$, with durations defined by a function $l: T \rightarrow \mathbf{Z}^+$, resource requirements $R_i: T \rightarrow \mathbf{R}_0^+$, and resource bounds B_i for $1 \le i \le k$, and an overall deadline $D \in \mathbf{Z}^+$; find (does there exist) a schedule $\sigma: T \rightarrow \mathbf{Z}_0^+$ such that

$$\sigma(t) + l(t) < D \quad \text{for all } t \in T \tag{1}$$

$$\sum_{\{t \in T \mid \sigma(t) \le j \le \sigma(t) + l(t)\}} R(t) \le B_i \quad \text{for all } 0 \le i \le n \text{ and } 0 \le j < D \tag{2}$$

where $\mathbf{Z}^+$ is the set of positive integers and $\mathbf{R}_0^+$ is the set of reals ≥ 0.

Under this notation, the set T defines the tasks that need to be scheduled. The tasks can be scheduled to start at any integral value of time between zero and the overall schedule deadline D. The resource requirements are defined by the functions R_i, which associate a real value with each task for each resource i. The resource bounds B_i defines the capacity of each resource. The function σ defines a schedule by assigning to each task an integral start time. Equations 1 and 2 guarantee that this schedule satisfies the overall deadline and the resource capacity bounds, respectively. However, this representation does not provide any mechanism for handling perturbations in the task durations, since only a single integer length is defined for each task by the function l.

The time constrained schedule representation extends this notation to the probabilistic case by assuming that the task length function returns an assumed duration of the activity. In general, one can define a family of mappings from the probability distribution for the task durations to an assumed duration for scheduling by

$$l_p(t) = \min \{z \in \mathbf{Z} \mid Pr(X_t \le z) \ge p\} \text{ for } 0 \le p \le 1 \tag{3}$$

where X_t is a random variable equal to the duration of task t. This formula defines the assumed duration of a task t, with respect to a probability p, to be the minimum duration for which the probability of completing the task is at least p. P is called the probability threshold.

This approach accommodates random variation in the task duration by defining a window in which the task can execute. The size of this window is controlled by the parameter p. When $p = 1.0$, the window is set to the worst case execution time for each task. A value of $p = 0.5$ would set the window for each task to the median value of the duration probability distribution. When this schedule representation is used by an onboard executive, a task would never begin before its assigned start time, as defined the

function σ. If the actual duration of any task exceeded the window defined for that task, we can no longer guarantee that the resource and deadline constraints are satisfied without resolving a NP-hard problem. Therefore, at this time a schedule failure has occurred. Since the boundary conditions by which a schedule failure is determined by the fixed time windows, this approach to accommodating variable duration tasks is called the time constrained representation.

The time constrained approach provides a simple mechanism for a real time schedule executive to be able to determine when to initiate tasks, while determining if the schedule remains valid in light of the actual durations seen so far. However, the time constrained representation is fairly fragile in terms of its resistance to failure. It is easy to see that the probability of a task successfully completing within its window is just p. If we assume that the durations for the tasks are stochastically independent, the probability that all n tasks will complete within their windows is p^n. As $n \to \infty$, $p^n \to 0$.

Order Constrained Schedules

The fragility of the time constrained approach is due to the fact that the schedule is successful if and only if all the windows completely surround the actual duration of their tasks. There is no capability in this approach for the random variations to "average out." Even if all but one task use less than their allotted time, but the one task exceeds its window, a schedule failure will occur. In trying to develop an alternative representation which allows for increased flexibility by allowing the random variations to accumulate and average out, the technique of pert charting naturally comes to mind.

In a pert chart, the schedule is represented as a directed acyclic graph (DAG). The DAG is a graphical representation of the predecessor - successor partial ordering. There are two commonly used representation of the DAG, called "activity on node" and "activity on edge." This paper will use the "activity on edge" representation. In the "activity on edge" representation of a pert chart, the nodes or vertices of this graph are called events, and the edges are the tasks or activities. If the edges *e1* and *e2* are part of a directed path through the DAG, in that order, then the task associated with *e1* is a predecessor of the task associated with *e2*. Occasionally dummy tasks need to be added to the DAG to accurately depict all the predecessor/successor relationships. These are usually drawn as dashed edges. The earliest possible start of a task is maximum length of all the paths that lead up to the start node for the task, where the length of an edge in the path is just the corresponding task duration. As the schedule executive executes the schedule, the actual durations can be substituted for the assumed durations for each task. This has the effect that a task can start only when all of its predecessors are finished.

With this idea as the basis for the order constrained approach, two questions need to be answered. How is the original resource constrained scheduling solution converted into a DAG, and how does the executive determine if the schedule is still valid based upon the DAG and the actual durations so far?

To illustrate the problems associated with creating a DAG from the resource constrained scheduling problem, consider the allocation of resource i as shown in FIGURE 1. In this figure, the horizontal axis represents

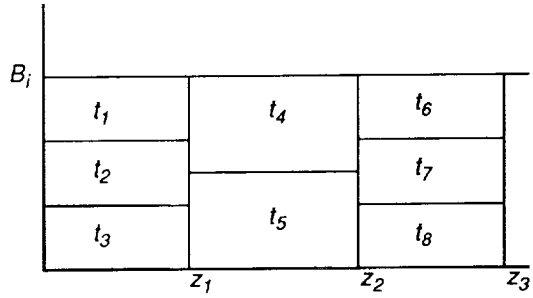


FIGURE 1 Resource timeline

time, and the vertical axis represents the allocation of resource i to the various tasks. In this example, t_1, t_2, t_3, t_6, t_7, and t_8 each have a resource requirement of 0.33 B_i. Tasks $t_4$and t_5 each have a resource requirement of 0.5 B_i. The time constrained approach guarantees that the sum of the resource requirements of all simultaneously executing tasks does not exceed the resource bound. For example, the executive would never allow tasks 1, 2 and 5 to execute simultaneously by ensuring that the windows for tasks 1 and 2 end before the window for task 5 begins. The problem for the order constrained approach is to define a partial ordering, implemented as a DAG, which accomplishes the same goal.

One straightforward way of accomplishing this goal is to define the partial order relation $<\bullet$ by

$$t_i <\bullet t_j \quad \text{iff} \quad \sigma(t_i) + l(t_i) \leq \sigma(t_j) .$$

In other words, this means that task t_1 precedes task t_2 if, and only if, t_1 is scheduled to finish at or before the scheduled start of task t_2. This in general will create more predecessor / successor relationships than are necessary, but it is a simple matter to go through and remove the redundant relations. FIGURE 2 shows the pert chart DAG which results from applying this procedure to the schedule in FIGURE 1.

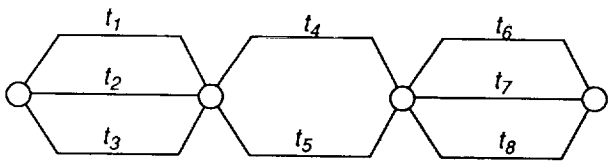


FIGURE 2 Pert DAG induced by resource constraints

While it can readily be seen that this partial ordering of the tasks will ensure that the resource capacity constraints are not exceeded, it can also be seen that it is overly constraining. For example, once tasks 1 and 2 complete, task 4 can be safely initiated since the resources required by tasks 1 and 2 are more than enough to satisfy task 4's requirement. Repeated application of this logic will eventually reduce the pert graph in FIGURE 2 to the graph shown in FIGURE 3.

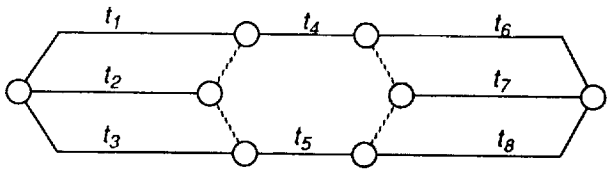


FIGURE 3 Reduced Pert DAG

Formally, then an order constrained schedule as a solution to an RCS problem is defined to be a partial ordering $<\bullet$ of the tasks in T such that:

$$\text{length}(\lambda) \leq D \quad \text{for all paths } \lambda \text{ in } <\bullet \tag{4}$$

$$\sum_{t \in S} R_i(t) \leq B_i \tag{5}$$

for all i, and for all $S \subseteq T$ such that
$t_1, t_2 \in S \Rightarrow (t_1, t_2) \notin <\bullet$ and $(t_2, t_1) \notin <\bullet$

Equation 4 is the revised constraint that guarantees that the partial ordering satisfies the overall deadline requirement. Equation 5 ensures that any set of tasks that might execute at the same time does not exceed the capacity of any resource.

One final question that needs to be addressed is how to calculate the length of a path through the pert network. Obviously, the length of the path should be the sum of the lengths of the individual tasks, but what value do we use for the length of the tasks, since we are assuming that these values vary? The a priori assumption, at schedule build time, is a task length based upon a probability threshold p, just as in the time constrained case. After the completion of the schedule, the a posteriori value of the task lengths is just the observed actuals. But what about during the execution of the schedule, when there are some actuals, and some unknowns? One could just use the a priori assumed lengths for the unknown durations. However a more general approach is to define a second probability threshold q, with $0 \leq q \leq p$. This defines a new length function l_q. The parameter q controls the amount of pessimism about the ability to recover when the actual execution is behind the a priori schedule. When $q = 0$, the executive will not declare a failure as long as there is some possibility of completing the schedule within its overall deadline by assuming that all remaining tasks will complete in their best case, or minimum durations. When $q = p$, the assumption is that the remaining tasks will complete in no less time than the a priori assumed durations. In either case, when the decision is made that the tasks will no longer complete by the overall deadline according to the current schedule, a failure is declared.

For a given partial order over the tasks of T, it is possible to calculate the length of the longest path, based upon the l_q length, and starting at the end node of each task t. If the actual end time of task t is later than $D - \max(l_q(\lambda))$, where λ is any path starting at t, then at least one path through task t will have a path length greater than D.

Therefore it is possible to precompute a deadline for each task by which time it must the task must complete in order for the schedule to meet the overall deadline in light of the actuals so far.

This suggests that it is possible to combine the two schedule representations into one. The combined representation consists of a partial ordering of the tasks of T, the window start times defined by the function $\alpha(t)$, and the window end time defined by $\Omega(t)$. For a time constrained approach, the partial order is empty, and the window start and end functions are defined by:

$$\alpha(t) = \sigma(t) \tag{6}$$

$$\Omega(t) = \sigma(t) + l(t) \tag{7}$$

For an order constrained approach, the partial is determined as described above, and the window start and end times are defined by:

$$\alpha(t) = 0 \tag{8}$$

$$\Omega(t) = D - \max_{\lambda \in P} (l_q(\lambda)) \tag{9}$$

where P is the set of all paths starting at the end node of t.

The job of the onboard schedule executive is to find all tasks that have no unfinished predecessors. Once the start window has been reached for these tasks, they are initiated. If any currently executing task fails to finish by its window end time, the schedule has failed and must be repaired by reinvoking the scheduler. It is fairly easy to see that the job of this onboard executive is tractable in the sense that it can be completed in a polynomial order of the number of tasks.

Development of Robust Schedules

Armed with this model of a flexible schedule representation than can accommodate some measure of perturbations during its execution, it is possible to define a method for using a deterministic scheduling system like COMPASS to build and manage robust schedules.

Since resource constrained scheduling is a NP-hard problem, COMPASS uses a mixed initiative dialog to generate feasible schedules that satisfy the user defined requirements.[2,3] Extending COMPASS to handle uncertain requirements, in particular probabilistic task duration, should therefore consist of adding commands to allow the user to interactively control the risk and uncertainty inherent in a particular schedule. Specifically, the user must be able to view, analyze and modify the risk and uncertainty inherent in a particular schedule. Analysis of a given schedule can be performed by performing Monte Carlo simulation of a large number of possible schedule executions to determine the MTTF of the schedule. If either the MTTF or the number of tasks the fit in the schedule is unacceptable, the user can adjust the a priori duration probability threshold and reschedule the tasks.

Conclusions

By combining fixed time windows with a pert style precedence graph, it is possible to build a schedule representation that can be executed and monitored by an automatic schedule executive in tractable way. Given that the durations of the scheduled tasks are not deterministic, but instead are represented by probability distributions, it is possible to identify probability threshold to control the a priori durations to use for scheduling and a posteriori limits to be monitored against. Given the probability distributions of the task durations, it is possible to perform a Monte Carlo analysis to determine the MTTF of a given schedule.

Acknowledgments

This research was supported in part by NASA under NAS9-17885.

References

1. Garey, M. R. and Johnson, D. S., *Computers and Intractability*, W. H. Freeman & Co., San Francisco, 1979.

2. Fox, B. R., *Mixed Initiative Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.

3. Fox, B. R., *Non-Chronological Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.

COMPASS
An Ada Based Scheduler

Mary Beth McMahon
Mc Donnell Douglas Space Systems Company
16055 Space Center Boulevard
Houston, TX 77062

Chris Culbert
Software Technology Branch
NASA - Johnson Space Center

Abstract: COMPASS is a generic scheduling system developed by Mc Donnell Douglas and funded by the Software Technology Branch of NASA-Johnson Space Center. The motivation behind COMPASS is to illustrate scheduling technology and provide a basis from which custom scheduling systems can be built. COMPASS was written in Ada to promote readability and to conform to DOD standards. COMPASS has some unique characteristics that distinguishes it from commercial products. This paper discusses these characteristics and uses them to illustrate some differences between scheduling tools.

Introduction

COMPASS is an automated scheduling system which is highly interactive. It allows the scheduler to control the placement of activities on the schedule while maintaining the integrity of the schedule by adhering to resource and temporal constraints. It is generic in the sense that it was not developed for any specific problem domain. Rather, any scheduling problem that can be described by activities, resources, and their constraints can be modelled using COMPASS.

The motivation behind COMPASS was to provide scheduling technology to the various NASA groups working on scheduling problems. It was written in Ada to promote readability and reusability. It is public domain software. Anyone working on a government project may obtain a copy of the executable and the source code to read or modify as needed. It is very portable and currently runs on a variety of machines. To support a larger number of users, two versions of COMPASS were produced: a graphical version and an ASCII version. The graphical version is based on X-Windows and runs stand-alone on a SUN3/ SUN4, SPARC station, IBM RS6000 and DEC VAX. In the client/server mode, it can be run over internet using an Apollo or a MacIntosh to display the image while executing on any of the stand-alone machines. For those with minimal computing resources, an ASCII version is available which prompts the user for commands. This version has the same scheduling power as the graphical version.

This paper will describe some scheduling concepts and will explore traits of COMPASS that distinguish it from commercially available products. This knowledge is useful in analyzing scheduling problems and determining which tools and algorithms best fit a particular scheduling need. The traits discussed in this paper also serve as a measure by which to compare many of the commercial scheduling tools currently available.

Data Representations

There are four primary data objects that COMPASS uses to describe a scheduling world: Activities, Resources, Conditions and Time. Activities are the items which must be scheduled. Resources are the objects which must be present in order for the activity to be performed. These include such things as tools, electricity, and people. Conditions describe the state of the scheduling world. Activities may require that certain conditions exist before they are scheduled. Conditions may include things such as in-orbit, daytime, Phase III, etc. Time is used in all scheduling packages; however, the way it is represented affects the largest and smallest time units that may be used by the problem.

The Activity Data Structure

COMPASS provides thirteen fields to describe an activity. All but two are optional. This allows the user to create a data structure that closely fits the scheduling problem. The two mandatory fields are the activity name and the duration of the activity. The following is a list of all of the fields of the activity data structure:

Name - *string - this is used as the unique id.*
Duration - *time type.*
Priority - *integer.*
Earliest Start - *time type.*
Latest Finish - *time type.*
Predecessors - *list of activity names that must precede this activity.*
Successors - *list of activity names that must follow this activity.*
Temporal Constraints - *timing constraints between the start or finish of one activity and the start or finish of another. There are four types of relations that may be represented: start/start, start/finish, finish/start, and finish/finish.*
Nonconcurrent Activities - *list of activity names that may not occur the same time as this activity.*
Preferred Intervals - *list of times which it is preferred that the activity occur. These times must fall between the earliest start and latest finish.*
Excluded Intervals - *list of times between the earliest start and latest finish in which the activity may not occur.*
Resource Requirements - *names of resources and their quantities required by this activity.*
Condition Requirements - *names of conditions and the times which they must exist in order for this activity to be scheduled.*

Most scheduling tools support the following fields: name, duration, priority, earliest start, latest finish, resource requirements, predecessors, successors and to some degree temporal constraints. The other fields are unique to COMPASS and have been included as optional fields in response to actual scheduling problems. Due to the modularity in the design of the activity data structure, it is trivial to add new fields to the activity. The scheduling engine is easily modified to conform to the requirements of the new field.

The Resource Data Structure

A resource is typically represented by the resource name and a quantity for a given time period. COMPASS uses this resource structure is used in three places- in the initial and net resource availability description and in the activity data structure to describe which resources are needed and in what quantities.

There are several characteristics of resources that must be considered when describing a resource need. Resource descriptions may be broken down into two types: Piecewise Constant and Piecewise Linear. Piecewise constant resource descriptions allow the use to request a quantity over an interval of time. More complex piecewise constant descriptions allow several quantities over several intervals of time. Piecewise linear descriptions specify a rate of consumption or production. An example might be the consumption of water - an activity might use a quart of water per minute for 1.5 hours. The closest approximation to this using piecewise constant is to describe it using "stair steps". However, this is more difficult to read and understand and is also not as accurate. A piecewise linear description would be a straight line whose second endpoint is 90 quarts lower and 1.5 hours later from its first endpoint.

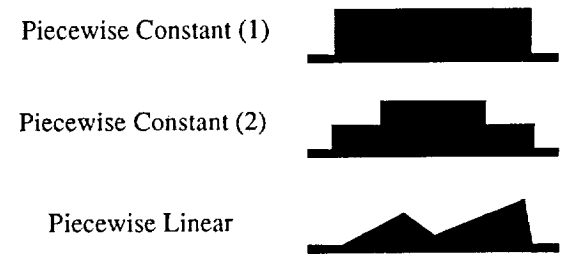


Figure 1. Resource Descriptions

Another characteristic of resources to consider is that they may be assignable, consumable or producible. An assignable resource is one that is used for the duration of the activity and then returned. A consumable resource is one that is consumed during the resource; therefore, when the activity finishes, that resource is no longer available. A producible resource is one that is produced by an activity.

It does not exist before the activity, but after the activity is under way the resource is available to be used by other activities. An example of an activity that both produces and consumes resources is the electrolysis of water. Water is consumed (at a rate) by the activity while hydrogen and oxygen are produced.

Most PC products support only the assignable resources and the first instance of piecewise constant resource descriptions. Larger software products usually only support assignable resources and both instances of piecewise constant resource descriptions. COMPASS supports all of the above types of resources and resource descriptions.

The Condition Data Structure

A concept that is not commonly supported by most scheduling tools is the ability to schedule activities only when certain conditions exist. COMPASS provides this capability using a data structure called Conditions. Conditions are used to describe the state of the world. Conditions are propositions whose values change between true, false and undefined over time. A condition is represented by a name and the intervals of time when the condition is true, false or undefined. If an activity requires that a certain condition be true, then the scheduling algorithm only schedules that activity during the span of time when the said condition is true.

Figure 2. Condition Representation

The Time Data Structure

Most scheduling packages use calendars to describe the times when resources are available and dates to specify when activities should begin and end. Typically, the smallest unit of time is one minute and the largest unit of time one year. Dates can range from the late 1900's to the early 2000's.

COMPASS supports date representations and relative time representations. Relative time allows the user to specify time requirements in relative measures. For example, the earliest start for an activity might be two weeks. This means that the earliest start time for the activity is two weeks after "time zero", where time zero is an anchor which does not correspond to a particular date. This allows the user to schedule activities two weeks before launch or one week after launch, without knowing exactly when launch is. Relative dates may be positive (eg. Launch plus one week) or negative (eg. Launch minus 3 days).

Characteristics of Scheduling Tools

There are three characteristics of a scheduling process which affect the way the schedule is built and the quality of the resulting schedule. These characteristics have minimal support, if any, in most commercial products; however, they are fully supported by COMPASS.

The first characteristic to look for is whether or not the scheduling tool is interactive. Can the scheduler affect the placement of activities when the schedule is being built? With most commercial products, all of the data is entered and then the schedule is created automatically placing each activity on the schedule as early as possible. COMPASS allows the user to put activities on the schedule individually or in groups. They can be placed as early as possible or as late as possible or at any feasible time in between. Activities are easily unscheduled and rescheduled to respond to various scenarios.

Another characteristic COMPASS supports is that of incremental scheduling. This allows the user to build a schedule in one activity at a time. It is not necessary to have all of the activities defined before starting to build a schedule. Once an activity is placed on the schedule, adding new activities does not interfere with those already on the schedule. That is, inserting an activity into the schedule does not change the starting times of any of the activities already on the schedule. This allows a user to make late additions to the schedule without affecting already scheduled activities. It also allows the user to place high priority items at preferred times with the guarantee that lower priority items will not move them from their slots.

A third characteristic is COMPASS supports non chronological scheduling. Non chronological scheduling allows the user to select any time in the future to place an activity. In chronological scheduling, a user starts at a designated time, schedules any activities that can be scheduled at that time, then moves forward in time until an event occurs which allows other activities to be scheduled. This may be compared to planning a weeks worth of activities. The non chronological method would provide a calendar on which you could see the entire week ahead of you and write down which activities you wish to do in any order on the days you wish to accomplish them. The chronological approach would require that you start writing down on Monday, determine which activities to do, write those down then move on to Tuesday, look at what is left to do, write those down then move to Wednesday, etc. until all of the activities have been scheduled. You could not say in advance that "I will do this Friday at 9:00", rather you would have to wait until you have scheduled everything from Monday to Thursday night, then Friday if it is still left to be done, then you can scheduled it. In this example, I moved forward one day at a time. In actuality, time is moved forward by the smallest increment or by events.

The Scheduling Algorithm

Another distinguishing characteristic of COMPASS is the scheduling engine. Most schedulers are based on the Critical Path Method (CPM) algorithm. CPM is a chronological based scheduling algorithm. CPM schedules each activity at its earliest starting time. The earliest starting time of each activity is determined by the latest completion time of all of its predecessors. CPM does not take into account any of the resource constraints. The result is a scheduling which finishes in the shortest amount of time, but oversubscribes resources. This type of scheduling algorithm is good for estimations and loading studies. A loading study is used to determine the number of resources needed to complete a schedule by a given date. But with most scheduling problems, the number of resources are limited. Therefore, oversubscribing the resources produces an infeasible schedule.

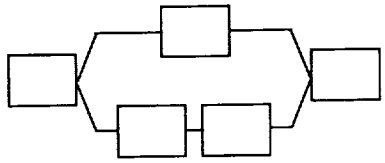


Figure 3. Critical Path Method

To alleviate the problem of oversubscription, most CPM tools also provide resource leveling. This technique shifts activities forward in time until the resources are available.

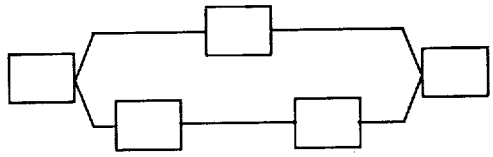


Figure 4. CPM with Resource Leveling

COMPASS does not use CPM. It uses a scheduling algorithm which takes into account both the temporal constraints and resource constraints. Given an activity, it determines all of the feasible times that the activity may be scheduled. Then the user provides some direction on the placement of the activity. This algorithm allows the user to select the order in which activities are placed on the schedule, as well as influence the placement of each as it is being scheduled. The feasible intervals of time in which an

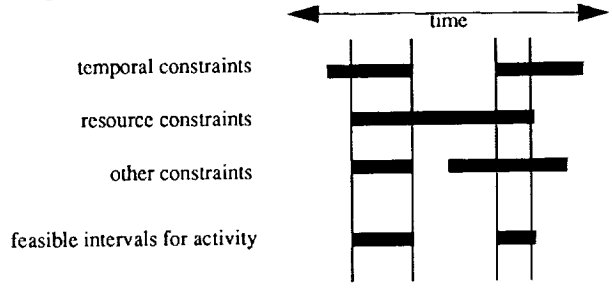


Figure 5. Computing Feasible Intervals

activity may be scheduled is determined by calculating all of the intervals of time that are feasible for each constraint and then taking the intersection. This gives the flexibility of choosing which constraints to enforce. It also provides

the user with information on all feasible times that the activity may be scheduled and allows the user to select the placement.

Another distinguishing feature of the scheduling algorithm used by COMPASS is that it allows the user to constrain both the temporal and resource constraints simultaneously. Most commercial products will create a schedule abiding by the timing constraints and oversubscribing the resources where necessary. Then if the user wishes, resource leveling can be applied. There are two ways resources can be leveled. First the activities can be moved forward in time to the point where enough resources are available to satisfy the activity. This causes the successors of this activity also to be moved forward in time also. Second, the quantities required by the activity can be reduced and the duration increased proportionally. This kind of resource leveling produces two potentially undesirable results. First, it changes the quantities and durations of activities. In some instances, this would not produce feasible results. Second, it may push activities forward in time past their deadlines. If there are hard limits on both time and resources CPM and resource leveling will not produce feasible schedules, because on an oversubscribed problem it will violate either the timing or resource constraints.

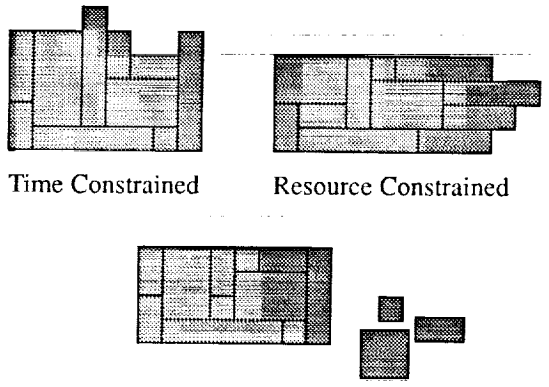


Figure 6. Three Ways to Handle Oversubscription

COMPASS will not violate the timing or resource constraints. Instead, it will inform the user that the activity cannot be scheduled. One reason for taking this approach is that most space related scheduling problems have hard limits on both the time and resources. For example, when scheduling space station or shuttle activities, the duration of the mission is fixed in advance and the number of resources is limited.

Conclusions

COMPASS is unique in that it provides the user more control over the placement of activities than most commercial products do. This is due to its unique scheduling algorithm which determines all of the feasible intervals of an activity then allows the user to affect the placement of it

COMPASS is also unique in that it will constrain both the resources and temporal relations simultaneously. This addresses those scheduling problems which have a fixed time to complete and fixed resources.

Acknowledgments

This development of COMPASS and related scheduling research is supported by NASA under NAS9-17885.

References

1. Fox, B. R., *Mixed Initiative Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.

2. Fox, B. R., *Non-Chronological Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.

OCCUPATIONAL SAFETY CONSIDERATIONS WITH HYDRAZINE FUELS

H.J. Clewell
Toxicology Division
Armstrong Laboratory
WPAFB OH 45433

T.S. Haddad
Toxicology Division
Armstrong Laboratory
WPAFB OH 45433

M.E. George
Toxicology Division
Armstrong Laboratory
WPAFB OH 45433

J.N. McDougal
Toxicology Division
Armstrong Laboratory
WPAFB OH 45433

M.E.Andersen
Toxicology Division
Armstrong Laboratory
WPAFB OH 45433

ABSTRACT

A simple pharmacokinetic model and a specially designed dermal vapor exposure chamber which provides respiratory protection were used to determine the rate of penetration of hydrazine and 1,1-dimethylhydrazine (UDMH) vapor through the skin of rats. Parameters for the pharmacokinetic model were determined from intravenous and inhalation exposure data. The model was then used to estimate the skin permeation coefficient for hydrazine or UDMH vapor from the dermal-vapor exposure data. This analysis indicates that UDMH vapor has a relatively high permeability through skin (0.7 cm/hr), a value somewhat higher than was obtained for hydrazine by the same procedure (0.09 cm/hr). Based on these skin permeability results, a skin-only vapor exposure limit giving protection equivalent to the inhalation Threshold Limit Value (TLV) could be calculated. The current TLV's for UDMH and hydrazine are 0.5 and 0.1 ppm, respectively. The corresponding skin-only TLV equivalents, for personnel wearing respiratory protection, are 32 ppm for UDMH and 48 ppm for hydrazine. Should the proposed lowering of the TLV's for these compounds to 0.01 ppm be adopted, the equivalent skin-only TLV's would become 0.64 ppm for UDMH and 4.8 ppm for hydrazine.

INTRODUCTION

Aerozine-50, a 50:50 by weight blend of UDMH and hydrazine, is an important fuel used in the Titan series missile. UDMH and hydrazine are caustic liquids which produce severe chemical burns. Therefore, personnel working with these fuels must wear protective clothing suitable to prevent liquid contact with the skin or eyes. In addition, inhalation of the fuel vapors has been shown to produce both acute and chronic toxicity, including carcinogenicity[1]. As a result, the American Conference of Government Industrial Hygienists (ACGIH) currently recommends TLVs of 0.1 ppm for hydrazine and 0.5 ppm for UDMH, and is considering lowering the TLVs for both chemicals to 0.01 ppm[2]. Thus respiratory protection is also necessary whenever significant concentrations of vapor may be present.

Because of uncertainty concerning the significance of dermal exposure to fuel vapor, it has often been the practice in the past to require that personnel working with these fuels wear a cumbersome, self-contained ensemble to provide full-body protection against vapor. This self-contained ensemble seriously degrades performance, so there is interest in establishing when protection of the skin from fuel vapors is needed, and when inhalation protection and prevention of liquid contact will suffice. Other than direct irritation of the skin, which occurs at levels well above the TLV, the principal issue concerning dermal exposure to fuel vapor is the potential for systemic toxicity due to penetration of vapor through the skin as compared to inhalation. The purpose of the current study was to estimate a dermal equivalent of the permissible hydrazine inhalation exposure level in order to establish realistic guidelines for ensuring the personal safety of individuals potentially exposed to hydrazine or UDMH vapors.

METHODS

Details of the experimental methods have been reported previously[3,4], so only a brief description is provided here. For the dermal vapor exposures, rats were exposed to hydrazine or UDMH vapors in a specially designed chamber[3] which provided respiratory protection by the use of masks, but allowed exposure of the whole body, which had been closely clipped of fur, to the chamber atmosphere. The same chamber was used for the inhalation exposures, but the use of masks was omitted and the fur was not clipped. Male Fischer 344 rats (200 to 280 g) were trained to wear a harness and, for the dermal exposures, latex face mask prior to the exposures. Twenty-four hours before the exposure, jugular cannulas were surgically implanted and exteriorized at the back of the neck. Just prior to the exposure, six rats were placed in the chamber and the cannulas were routed outside the chamber for blood collection during the exposure. For intravenous exposures, hydrazine or UDMH was introduced in a saline vehicle. Blood samples were drawn through a jugular cannula at 0.5, 1, 2, and 4 hours after injection. Analysis of blood samples involved

derivitization with chlorobenzaldehyde[3] or acetone[4], followed by gas chromatography.

A relatively simple one-compartment open pharmacokinetic model was used to relate the measured blood concentrations during exposure to the total amount of chemical (hydrazine or UDMH) in the animal and the total amount metabolized. Two processes were incorporated in the model: slowly reversible chemical reaction with components of the blood and tissues, and irreversible clearance of free chemical by a saturable process. Incorporation of these two processes into the model was necessary to provide a coherent description of all of the data from the three exposure routes. The total amount of unreacted chemical in the animal (Afree) at a given time is determined by integrating the following equation with initial condition Afree = 0, for the vapor exposures, or Afree = dose (mg), for the intravenous exposures:

dAfree/dt = Rskin + Rinh - Rx - Rclear
where:

Rskin = rate of penetration through skin

= P * A * Cdermal
Rinh = rate of inhalation
= Qalv * Cinhal
Rx = rate of reaction
= Kr * Cfree * Vd - Kd * Ax
Rclear = rate of clearance
= Vmax * Cfree / (Km + Cfree)

In these equations, P is the skin permeation coefficient (cm/hr), A is the skin area (cm^2), Cdermal is the chamber concentration for the dermal vapor exposures (mg/ml), Qalv is the alveolar ventilation rate (L/hr), Cinhal is the chamber concentration for the inhalation exposures (mg/L), Cfree is the concentration of unreacted chemical in the blood (mg/L), Kr is the rate constant for reaction with blood and tissue components (/hr), Vd is the volume of distribution in the animal (L), Kd is the rate constant for reversal of the reaction with blood or tissue (/hr), Ax is the amount of reacted chemical (mg), Vmax is the maximum clearance rate for free chemical (mg/hr), and Km is the Michaelis-Menton constant for the clearance process (mg/L). Two major assumptions of the model are that the rate constants for reaction are the same in blood and tissues, and that the hydrazone derivative procedure effectively reverses the reaction with blood components so that the measured concentration is the sum of free and reacted chemical:

Cmeasured = Cfree + Ax/Vd

Qalv and A were calculated on the basis of empirically derived relationships to body weight, with typical values of about 6.7 L/hr and 260 cm^2, respectively. The volume of distribution was set to total body water (65% of body weight). The kinetic constants Kr, Kd, Vmax, and Km were established by iterative fitting of the inhalation and intravenous exposure data sets. Although these parameters are highly correlated, their affects on the behavior of the model were sufficiently distinct to estimate each of the parameters relatively independently. The final step was to estimate P from fitting the dermal vapor exposure data. The numerical integration and non-linear parameter estimation was performed with SIMUSOLV (Mitchell and Gauthier Associates, Inc., Concord MA) on a VAX 8530.

RESULTS

The model was able to provide a reasonably good representation of the intravenous and inhalation data for UDMH (Figs. 1 and 2) using parameter values of Kr=11/hr, Kd=1.9/hr, Vmax=8.7 mg/hr, and Km=0.6 mg/L. Although the model somewhat overestimates the blood concentrations at early times in the inhalation exposure, it predicts the correct steady-state behavior and performs remarkably well at describing the complex intravenous kinetics. The value of the skin permeation coefficient that yielded the best agreement between the model and the dermal vapor exposure data was 0.7 cm/hr (Fig. 3).

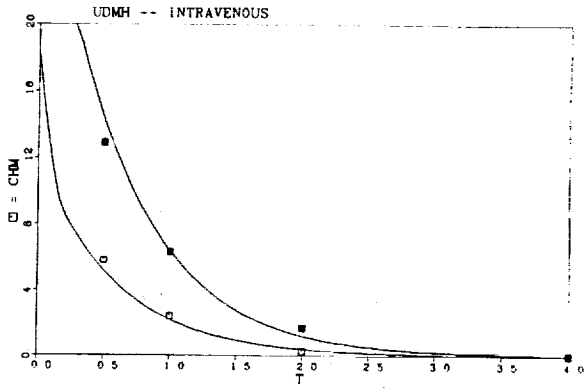


Figure 1. Model-Predicted (curves) and Experimental (symbols) Blood Concentrations of UDMH for Intravenous Doses of 24 mg/kg (upper curve, solid symbols) and 12 mg/kg (lower curve, open symbols).

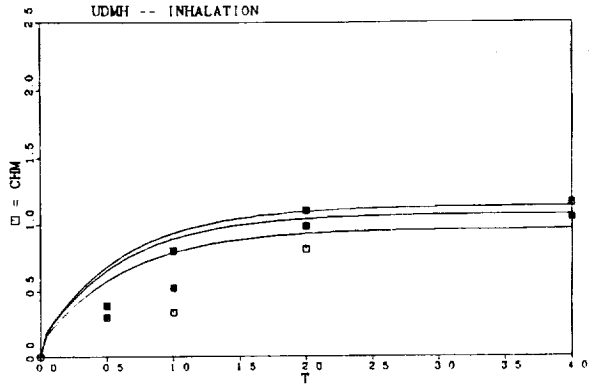


Figure 2. Model-Predicted and Experimental Blood Concentrations of UDMH for Inhalation Exposures at 108 ppm (upper curve, solid symbols), 104 ppm (middle curve, crossed symbols) and 95 ppm (lower curve, open symbols).

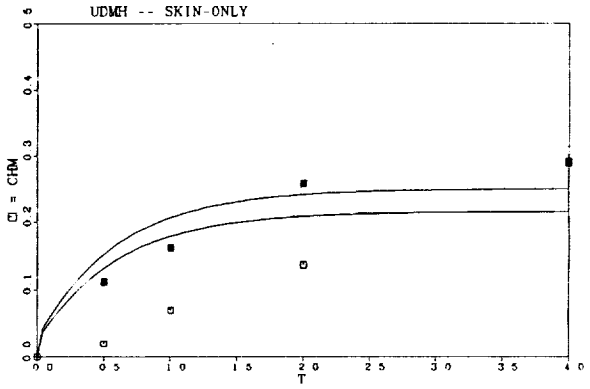


Figure 3. Model-Predicted and Experimental Blood Concentrations of UDMH for Skin-Only Vapor Exposures at 1028 ppm (upper curve, solid symbols) and 895 ppm of UDMH vapor (lower curve, open symbols).

An earlier analysis for hydrazine[5] obtained a value of 0.06 cm/hr, but that analysis used a very different pharmacokinetic model. To assess the impact of the change of model, the model described above was used to re-evaluate the hydrazine data. The new model was able to provide a general description of the hydrazine data as well (Figs. 4 and 5), using parameter values of Kr=.031, Kd=.055, Vmax=3.1, and Km=50. The only other change required was to restrict the volume of distribution to the blood volume (5% of body weight). The value of P determined with the new model was 0.092 (Fig. 6), in good agreement with the earlier value.

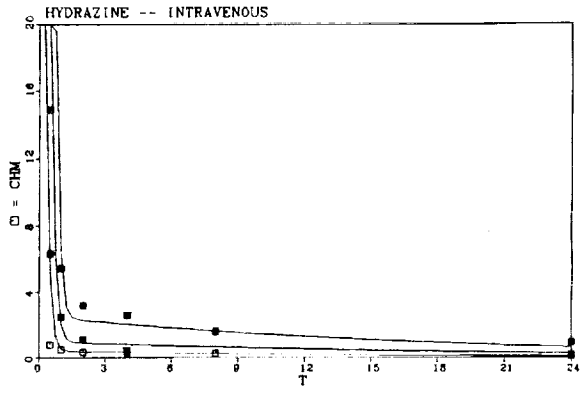


Figure 4. Model-Predicted and Experimental Blood Concentrations of Hydrazine for Intravenous Doses of 12 mg/kg (upper curve, solid symbols) and 6 mg/kg (lower curve, open symbols).

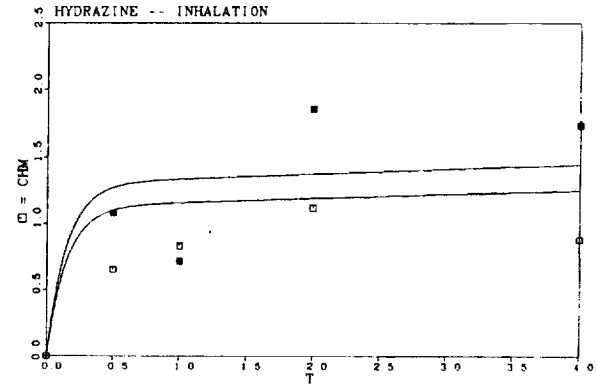


Figure 5. Model-Predicted and Experimental Blood Concentrations of Hydrazine for Inhalation Exposures at 10.7 ppm (upper curve, solid symbols) and 9.3 ppm (lower curve, open symbols).

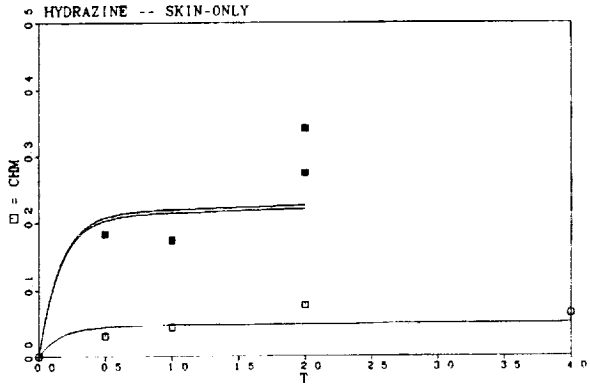


Figure 6. Model-Predicted and Experimental Blood Concentrations of Hydrazine for Skin-Only Vapor Exposures at 486 ppm (upper curve, solid symbols), 476 ppm (middle curve, crossed symbol), and 105 ppm (lower curve, open symbols).

Once we have established the skin permeation coefficients, we can calculate the relative importance of the dermal and inhalation routes of exposure for systemic toxicity. Assuming that the rat is a good model for skin absorption in the human, a measure of the rate of uptake of chemical through the skin is simply P*A, where A is the exposed skin area for humans, generally taken to be 2 m^2. The equivalent measure of uptake for inhalation is just Qp, which is commonly taken to be 800 L/hr. The ratio of inhalation to dermal uptake is then Qp/(P*A). Applying this formula to UDMH, the ratio of inhalation to dermal uptake is then 64, and the similar ratio for hydrazine, using P=0.09, is 487. Multiplying these ratios by the current TLVs, we obtain estimates for the equivalent skin-only exposure levels of 32 ppm for UDMH and 48 ppm for hydrazine. At the proposed TLVs of 0.01 ppm, the skin-only equivalents would be 0.64 ppm and 4.8 ppm, respectively.

DISCUSSION

The approach described above for estimating safe skin-only vapor exposure levels relies on a number of assumptions. The principal assumption is that skin permeation coefficients measured in rats are representative of human skin values. While there is some evidence that human skin is at least as good a barrier to chemical vapors as rat skin[6], the exact relationship has not yet been established for water soluble chemicals. The adequacy of the rat as a model for skin absorption in humans is therefore an important source of uncertainty which should be considered in deriving human exposure guidelines. Another key assumption in this approach is that the skin and lungs act only as routes of entry for a chemical whose effects are systemic. The method would be inappropriate for a chemical which had direct effects either on the skin or in the lungs at the concentrations of interest. In the case of hydrazine and UDMH there are no significant contact site effects at the level of the TLV.

Given the assumptions discussed above, this same approach can be generally applied to other toxic chemicals. Provided that skin permeation coefficients for the vapor are either known or can be determined, the calculation of the skin-only equivalent to a given inhalation exposure guideline is straightforward. This approach could provide a method for use by ACGIH or OSHA in assigning quantitative skin notations. A similar approach can also be used to determine appropriate personal protection for potential short-term exposures as a part of emergency response planning[7].

REFERENCES

1. Committee on Toxicology, Emergency and Continuous Exposure Guidance Levels Selected Airborne Contaminants, Volume 5. National Academy Press, Washington, D.C., 1985.

2. American Conference of Government Industrial Hygienists, Documentation of the Threshold Limit Values and Biological Exposure Indices. ACGIH, Cincinnati, Ohio, 1990.

3. McDougal, J.N., M.E. George, H.J. Clewell III, M.E. Andersen, and D.L. Pollard, "Dermal Absorption of Hydrazine Vapors" 1985 JANNAF Safety and Environmental Protection Subcommittee Meeting, CPIA Publication 436, Chemical Propulsion Information Agency, Laurel, Maryland, 1985.

4. Clewell, H.J., T.S. Haddad, T.E. Fazekas, J.N. McDougal, and M.E. Andersen, "Dermal Absorption of 1,1-Dimethylhydrazine (UDMH) Vapor" 1988 JANNAF Safety and Environmental Protection Subcommittee Meeting, CPIA Publication 485, Chemical Propulsion Information Agency, Laurel, Maryland, 1988.

5. Clewell, H.J., J.N. McDougal, M.E. George, and M.E. Andersen. "Occupational Safety Considerations with Hydrazine." Third Conference on the Environmental Chemistry of Hydrazine Fuels, AFESC-TR-88-000, 1988.

6. McDougal, J.N., G.W. Jepson, H.J. Clewell III, M.L. Gargas, and M.E. Andersen, "Dermal Absorption of Organic Chemical Vapors in Rats and Humans" Fundam. Appl. Toxicol. 14:299-308, 1989.

7. Clewell, H. J. "Exposure Guidelines for Materials Toxic by Inhalation: Search for a Rational Approach" 1990 JANNAF Safety and Environmental Protection Subcommittee Meeting, CPIA Publication 543, Chemical Propulsion Information Agency, Laurel, Maryland, 1990.

N93-11952

Title: Manager's Assistant Systems for Space System Planning

Authors: William L. Bewley, Robert Burnard, Gary E. Edwards, James Shoop

Abstract:

This paper describes a class of knowledge-based "assistant" systems for space system planning. Derived from technology produced for the DARPA/USAF Pilot's Associate program, these assistant systems help the human planner by doing the bookkeeping to maintain plan data and executing the procedures and heuristics currently used by the human planner to define, assess, diagnose, and revise plans. Intelligent systems for Space Station Freedom assembly sequence planning and Advanced Launch System modeling will be presented as examples. Ongoing NASA-funded work on a framework supporting the development of such tools will also be described.

SECTION II

AUTOMATION AND ROBOTICS

Session A1: AUTOMATION AND ROBOTICS PLANNING SESSION

N93-11953

REQUIREMENTS AND APPLICATIONS FOR ROBOTIC SERVICING OF MILITARY SPACE SYSTEMS

Dr. Otto C. Ledford Jr.
Director of Space Technology
PRC Inc.
222 N Sepulveda, Suite 1310
El Segundo CA

Maj. Rodney G. Bennett
Chief, Engineering Technology
Air Force Space Systems Division
El Segundo CA

ABSTRACT

The utility of on-orbit servicing of spacecraft has been demonstrated by NASA several times using shuttle-based astronaut EVA. There has been interest in utilizing on-orbit servicing for military space systems as well. This interest has been driven by the increasing reliance of all branches of the military upon space-based assets, the growing numbers, complexity, and cost of those assets, and a desire to normalize support policies for space-based operations.

Many military satellites are placed in orbits which are unduly hostile for astronaut operations and/or cannot be reached by the shuttle. In addition, some of the projected tasks may involve hazardous operations. This has led to a focus on robotic systems, instead of astronauts, for the basis of projected servicing systems.

This paper describes studies and activities which will hopefully lead to on-orbit servicing being one of the tools available to military space systems designers and operators. The utility of various forms of servicing has been evaluated for present and projected systems, critical technologies have been identified, and strategies for the development and insertion of this technology into operational systems have been developed.

Many of the projected plans have been adversely affected by budgetary restrictions and evolving architectures, but the fundamental benefits and requirements are well understood. A method of introducing servicing capabilities in a manner which has a low impact on the system designer and does not require the prior development of an expensive infrastructure is discussed. This can potentially lead to an evolutionary implementation of the full technology.

1. HISTORY OF SPACE-BASED SERVICING

Space-based systems are very valuable for many diverse applications. They are also very expensive to build and to place into orbit. Although satellites are designed to high standards and extensively tested on the ground, they have suffered from the range and rate of anomalies to be expected fro such complex systems.

For the first 15 years of space activity, troubled systems could be salvaged only by creative reprogramming from the ground. For example, a Tracking and Data Relay Satellite (TDRSS) was raised into a useful orbit using its attitude control thrusters following a problem with the IUS. If such a work-around could not be achieved, there was no alternative but to launch a replacement system.

Once man began gaining regular access to space, the alternative of direct action on the problems became possible. There have been a number of space missions which have been saved by corrective actions taken by astronauts performing Extra Vehicular Activities (EVA). The first of these was on the initial deployment of Skylab in 1973 when repairs were made to the solar panels and thermal shield which had been damaged during launch. Had it not been for this manual improvisation, the vehicle would have been unusable.

In 1984, an EVA repair mission from the shuttle replaced a failed attitude control module on the Solar-Max satellite and restored that vehicle and its payload to full operational status. An additional 6 years of valuable data was obtained as a result of this repair. There were plans to revisit this satellite in 1990 for a second servicing mission to recover it before re-entry, but the only available flight opportunity was pre-empted by the recovery of the Long Duration Exposure Facility.

A pair of communication satellites, Palapa and Westar, were stranded in low earth orbit due to failures in their boost propulsion systems. Although these satellites were not originally designed with provisions for orbital handling, an on-orbit recovery mission was carried out from the shuttle in 1985. The spacecraft were recovered by astronaut EVA operations, refurbished, and later relaunched.

The most recent example of a space mission being saved by on-orbit action is the deployment of the Gamma Ray Observatory (GRO) on STS-37. After the high-gain antenna failed to deploy, the crew performed an EVA which successfully freed a stuck boom by using a crank designed for ground operations. Without this corrective action, the $650 million spacecraft would have been unable to return useful data to earth.

The Soviets have also salvaged several missions by on-orbit repairs of their Salyut/MIR space stations. Problems they have corrected by EVA range from the release of jammed mechanisms to the replacement of portions of a fluid system, which involved cutting and welding tubing.

2. MILITARY INTEREST IN SPACE SERVICING

These demonstrations created an interest within the Department of Defense in the benefits of space servicing operations. The use of space-based systems has become an integrated part of military doctrine, and the number, complexity, and expense of these systems is projected to increase in the future. This growing military reliance upon space requires that the systems meet the required operational availability and be fielded and operated within budgetary constraints. It was recognized that extending the service life of space-based assets by correcting conditions which would otherwise terminate the mission could reduce the life cycle costs and increase the operational utility of military systems.

Military interest in space-servicing techniques was given a major stimulus by the Strategic Defense Initiative (SDI). The early SDI architectures featured large constellations of space-based weapons and surveillance systems. The potential supportability requirements associated with large numbers of complex and expensive satellites were identified as critical issues in the development of these systems.

3. DIVERGENCE OF DOD AND NASA REQUIREMENTS

Although DoD is following NASA's lead into orbital servicing and wishes to fully leverage NASA's experience and technology base in this area, there are significant differences between the needs and interests of the two agencies. NASA has developed a satellite servicing methodology and technology based upon astronauts performing EVA operations from the shuttle orbiter or, in future years, from the Space Station. As in the examples above, these operations take place in relatively low altitude earth orbits that can be reached by the shuttle orbiter.

Most of the prospective DoD candidates for servicing would be located in high altitude or high inclination orbits which could not be reached by the shuttle. In addition to military space assets being difficult to reach, many of them are located in environments are hazardous to astronaut operations, such as the radiation belts. It is also recognized that performing an EVA is inherently a high risk operation. If space-based servicing were to be incorporated as an operational tool, a potentially large number of remote operations might be required. It is prudent to reduce the potential risk to human life if a suitable alternative can be made available.

Another important difference between NASA and DoD requirements is the timeliness of response to problems. Most NASA systems are scientific satellites. The interruption of their data is an inconvenience that can generally be recovered from by later observation time. However, the critical defence missions of military satellites dictate that these systems be restored to full operational availability as quickly as possible after an anomaly. This may require timelines which are incompatible with the time required to schedule and launch a manned response but which can be met by a dedicated launch of a small unmanned system.

These requirements have led the DoD to emphasize a robotic approach utilizing expendable launch vehicles for the servicing of space assets rather than the astronaut EVA approach developed by NASA.

4. FAILURE MODES OF SPACE SYSTEMS

The requirements definition for a space-based servicing system begins with an examination of situations for which servicing might be an appropriate response. These failure modes can be classified into several broad categories. The first of these catagories is the failure of a component which results in the partial or complete loss of system functionality. These failures will almost always be of a random nature, since almost the entire mission life falls within the regime described by the flat or random failure region of a standard "bathtub shaped" reliability curve, as shown in Figure 1.

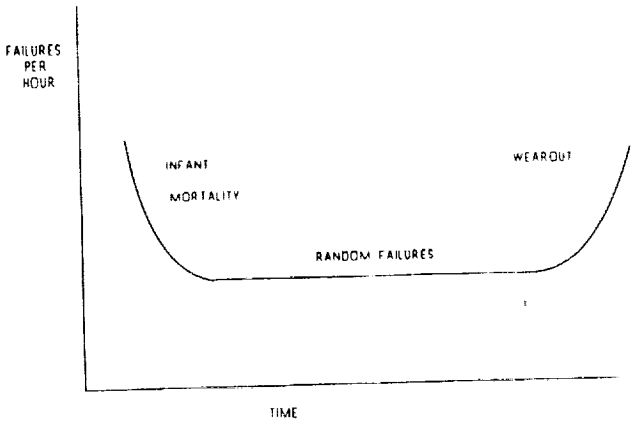


FIGURE 1. GENERIC RELIABILITY CURVE

There will be a relatively high rate of "infant mortality" early in the mission. With a few minor exceptions, the increase in failure rate associated with the wear out portion of the reliability curve is never reached.

Since these failures can occur at any time (although concentrated early in the mission) and often without warning, they are the most stressing in the time required for a response.

The second category is that of degradation in the performance of a component or subsystem. This type of failure has the effect of a gradual reduction in the performance margin of a system, sometimes reaching a threshold at which system functionality is lost. This usually gives ample time to plan a response, especially in cases where the rate of degradation can be well characterized. An example is degradation of solar cells due to radiation damage.

A third life-limiting category is depletion of consumables. It is somewhat similar to degradation in that it is a time- or cycle-dependent phenomena, but fundamentally different on that the performance of the system is not affected until the consumable is depleted, at which time functionality is lost. Consumption of fuel in a propulsion system is a typical example.

Another category where servicing might be beneficial is that of a system which has not failed, but has become outmoded. A capability of inserting upgraded technology into the basic system could extend the spacecraft's useful life, enhance its mission capabilities and lower life cycle cost.

5. RESPONSES TO FAILURE MODES

There are three general types of responses available to life limiting issues or failure modes. The first of these is to change the design. If a mission is limited by depletion of a consumable, adding a larger amount of that consumable to the original design is more efficient than supplying more once the system is on station, provided that the additional mass is within the available launch capacity. Additional margin can be built into systems which suffer from degradation, with the same proviso on weight limitations. Reliability of complex systems can be improved by providing both component level and subsystem level redundancy. However, there is a limit to the level of redundancy which can be provided before the potential failures arising from the increased complexity outweigh the benefits. Launch weight limitations can be a factor here as well.

A more sophisticated design change would be to change the function. Failure modes such as those associated with mechanical mechanisms can be designed out of the system or alternative approaches can be used for the overall system architecture. A subtle variant is to reconfigure the system by telemetry to "work around" anomalies.

These design approaches are normally used to maximize service life, and will continue to be the most appropriate first response for failure modes which can be clearly identified.

However, experience has shown that good design practice alone cannot eliminate on-orbit failures. The classical response to such failures is to abandon and replace the failed, degraded, or obsolete asset. This will continue to be the most appropriate response in some cases.

The emerging alternative to abandonment of failed assets is on-orbit servicing. This response has been demonstrated on an ad-hoc basis on systems which were not originally designed for servicing, but offers the greatest promise if it is a basic design feature of the system. Servicing is not a universal panacea, but merely another tool which the system designer can exploit to achieve the most responsive concept. The economic viability of servicing is dependent upon the cost, weight, and inherent reliability of the satellite, launch costs, and the nature of the failure. It must be recognized that designing for servicing is an additional requirement which can conflict with other design requirements and may entail cost and mass penalties. However, these penalties may be offset by producibility and testing benefits as well the benefits gained by servicing.

In each case, the choice of whether the potentially mission limiting feature should be responded to by design changes or by operational servicing is a question to be decided on the basis of lowest cost, with the answer tempered by technical capabilities. For military systems, operational needs and availability will also be a strong consideration and may outweigh other factors. In many cases the optimum solution will utilize both approaches, with the system designed to provide the maximum possible life and servicing providing an extension in capability beyond that constrained by engineering or initial launch weight.

6. KEY SERVICING TECHNOLOGIES

Our examination of the specific operations required to perform on-orbit servicing identified the technologies which must be developed to achieve this capability. We found these to be relatively few and well within current engineering practice. The most critical issue appears to be that of designing the systems for servicing.

As we evaluated the details of how to perform servicing of failed or degraded components it became apparent that current military design practice is not compatible with robotic on-orbit servicing. Systems are assembled by building outward from a tightly packed interior. Components are integrated on equipment platforms and are thus largely inaccessible. It is pointless to debate the levels of robotic technology required when these tasks could not be performed even by an astronaut with a complete tool kit.

The full benefits of on-orbit servicing can be achieved only if the spacecraft to be serviced is designed from its very inception with a large percentage of its systems modular and accessible. Robotic removal of these modules should also be a design consideration. This goal is best implemented by a spacecraft architecture featuring standard fittings, interfaces, and docking locations with the orbital replacement units (ORUs) located on an external frame where they are accessible.

An important consequence of this modular architecture is that it implies a radically new manufacturing, test, and assembly flow. Modules can be built up in parallel and tested independently by plugging into spacecraft bus simulators. Errors could be detected and corrected at a lower level with less impact than in the traditional buildup of equipment platforms into spacecraft.

The use of standard subsystem ORUs across several programs would enable an economy of scale to be achieve which could substantially reduce unit costs. The greater experience base for each hardware item would also allow the ultimate reliability rates to be improved over what they would be if a specific item were developed for each application. This module standardization and mass production also enhances the availability of production line spares. These spares and the ease of repair on the ORU level will reduce production delays on the system level.

The synergistic relationship between modularity and serviceability recalls the experience of the Solar Max. The on-orbit repair which returned it to service was possible not because the MMS spacecraft had been designed to be serviceable but because it was designed to achieve the production and assembly benefits of modularity. The repair was an unanticipated benefit of the resultant accessibility.

Our implementation analysis identified several technologies which are either enabling or enhancing to remote servicing missions. Three of these were judged to be key in that they must be demonstrated in order for on-orbit to be fully accepted as an operational concept. These are autonomous rendezvous and docking, robotic ORU replacement, and fluid transfer.

It is possible to fly a rendezvous and docking mission entirely under remote piloted control. However, an autonomous system would significantly reduce operational support requirements and costs as well as provide higher reliability and more efficient propellant usage. This technique is regularly used by the USSR in supplying their space station but has not yet been demonstrated by the US.

The location of many DoD assets in orbits with high radiation fluxes or other hazards makes it desirable to perform servicing robotically. Robotic replacement of ORUs by simple, well defined motion can be enabled by a standardization of docking locations and fittings. This can eliminate the requirement for highly capable robots with advanced sensing, manipulative, and cognitive capabilities. The man-in-the loop requirements will be minimized by the definition of the locations and motions to be executed. This autonomy reduces the criticality of the time delay in the feedback loop which has been seen as a limit to tele-operations.

There is a trade-off between the degree of structure provided in the environment and the level of robotic technology required. It appears best to begin with a simple, structured system which has a growth path to higher complexity. The servicing robot would operate in the supervised autonomy mode, carrying out a series of pre-programmed "macro" instructions, pausing between each operation for verification and

authority to proceed. Manual override and tele-robotic operation would be available for dealing with unanticipated situations. It is important to design the robot to fail in a safe and recoverable mode and not induce failures on either the servicer or the spacecraft being serviced.

The reduction of robotic operations to simple, well defined motions allows the robotic servicing subsystem to be very simple. This is a very important concept because this simplicity allows the robotic device to achieve a high level of reliability. On-orbit servicing can be economically viable only if the failure rates of the support system are extremely low. This simplicity also allows the robotic servicer to be developed with near-term technologies.

The connectors and procedures for on-orbit fluid transfer comprise the third key technology which must be demonstrated. The replenishment of propellants or other fluids was identified as one of the prime candidates for on-orbit servicing. DoD has a near term interest in hydrazine for refuelling a variety of spacecraft in both low and high altitude orbits, with a longer term interest in liquid cryogens for propulsion or space-based weapons. NASA is interest is in supplying superfluid helium to scientific satellites such as the Space Infrared Telescope Facility (SERTF).

Since refuelling can be implemented with minimal impact upon the spacecraft architecture, its potential benefits have been examined in several government sponsored studies. The most recent of these was performed for Air Force Space Systems Division Long Range Planning and Development Office. The study found two classes of satellites which could benefit from on-orbit refuelling. These were satellites in low earth orbit which had a basic propulsion requirement for drag makeup, and those in geo-synchronous orbit which have a base requirement for station-keeping. In both cases, the benefits of refuelling were derived more from the operational flexibility of greater maneuver capability than by addressing the basic propulsion requirement.

Analysis usually shows that the most cost-effective method of extending propulsion lifetime of satellites is to increase the size of the original fuel tanks, up to the limit imposed by booster capacity. If the remainder of the payload is of sufficiently high reliability and value, it can even pay to move to the next larger class of booster.

The problem of matching the fuel capacity to the rest of the system lies in the nature of the failure rates of the rest of the system. The design life of the system, which is normally used for sizing of subsystems, is not a hard limit unless it is set by exhaustion of a consumable. It is not even marked by a cluster of expected failures unless system lifetime is dominated by wearout failures. System lifetime is usually determined by random failures which occur at an even rate once the infant mortality period is passed. Thus the design life is a statistical concept denoting the point at which the predicted availability falls below the system allocation.

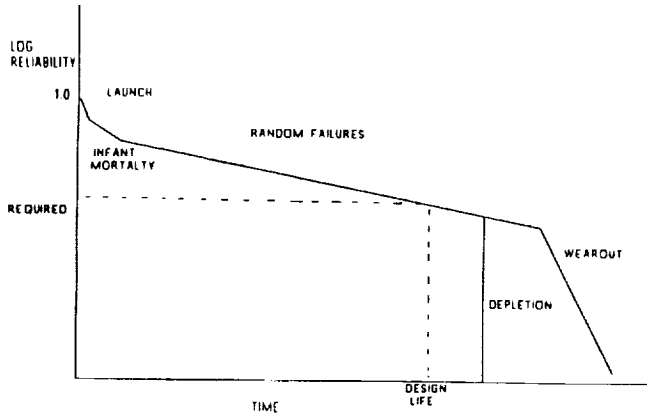


FIGURE 2. SYSTEM RELIABILITY OR AVAILABILITY

The probability that a space system will be operating satisfactorily follows the pattern shown in Figure 2. There is an immediate drop in reliability due to the chance of failure during launch and deployment. The availability then decreases with time during the early infant mortality period, with the rate of failure decreasing as the random failure region is reached. The failure rate increases again if the wearout region is reached. However, depletion of a consumable such as fuel will usually terminate system operation before wearout is reached. A nominal design life point is shown.

It is this statistical basis of the random failures that provides an economic incentive for refuelling. There will be systems which will reach the nominal design life without failure. If they are sufficiently far from a wearout mode limit they will be no more likely to fail during the next x years than they were during the first x years. Therefore, there is additional useful life which could be obtained by refuelling. The statistical chance that there might also be an unrecoverable failure early in the mission makes it unprofitable to oversize the fuel tanks to the point of requiring a larger booster.

The economics and utility of refuelling can thus be seen to be justified primarily on a contingency basis. It is those off-nominal cases where the system is required to make unanticipated maneuvers, or when the system continues to operate successfully

beyond the nominal design lifetime that benefit from the capability to provide additional fluids in orbit. These benefits can be potentially large in relation to the costs required to achieve them.

7. REQUIRED TECHNOLOGY DEMONSTRATIONS

Spacecraft fluids can be classified by physical properties and hardware technology requirements into five general categories.

Gases
Monopropellants (includes water)
Bipropellants
Superfluid Helium
All Other Cryogens

There is nothing unique to the space environment which affects gas transfer, so its technology is well in hand. Supply and handling systems for the other four categories have each been given some development attention, and vary to the degree of technical maturity.

The on-orbit transfer of hydrazine was demonstrated by an experiment in the shuttle payload bay in 1984. Hydrazine was transferred from one tank to another after a valve was opened by an astronaut. This resolved many of the issues associated with low-gravity fluid transfer, including adiabatic recompression. The Storable Fluid Management Experiment further resolved micro-gravity fluid handling issues by observing the behavior of water in a transparent tank in the shuttle mid-deck in 1985. There are still some issues to further explored, such as mass gauging and robotic operation of "zero-leak" connectors under micro-gravity conditions. However, the physics is well understood and only a demonstration is required.

NASA is current developing an on-orbit demonstration of superfluid helium transfer in the SHOOT (Superfluid Helium On Orbit Transfer) experiment. This will resolve many of the additional technical issues peculiar to this fluid such as the fluid transfer process and mechanisms, tank chill-down and venting operations in micro-gravity, and verification of thermal models for heat transfer within the tank due to mixing.

A substantial amount of hardware supportive of on-orbit fluid transfer has been built and tested on the ground. Several prototype valves and connectors suitable for robotic operation have been developed under contract to NASA-JSC. An automatic fluid interface system (AFIS) which could hold these valves and act as a coupler between the servicer and target spacecraft has been built under contract to NASA-MSFC.

Much of the remaining technical development required for an on-orbit fluid transfer system can be performed as part of an experiment within the shuttle payload bay. However, true confidence on the part of potential users will not be obtained until mating and fluid transfer is demonstrated on free-flying satellites.

There is very little technology work required for an autonomous rendezvous and docking system, as all the requisite technologies exist as components. What remains to be done is the system integration and demonstration. The control system operation can be partially verified by a test and evaluation program which begins with software simulation and evolves into hardware-in-the-loop simulations on air-bearing tables. However, the final full-up demonstration of responses in all six dynamic dimensions must be achieved as a free-flying space experiment. Although preliminary development of the sensor systems can be done on ground test ranges, a space flight experiment will be required to verify performance at operational ranges against realistic backgrounds.

The development and demonstration requirements for the robotic manipulator system parallel those for the docking control system. The control loops and end effector operation can largely be demonstrated on the ground in all aspects except those affected by micro-gravity. Manipulator arm dynamic response in a micro-gravity environment is important only for large, light assemblies such as those being developed for the Flight Telerobotic Servicer (FTS). A simpler, more compact manipulator as described above might have enough stiffness to be verified in ground tests. The arm dynamics can be assessed as part of an attached shuttle payload experiment.

The major dynamics issue which can be resolved only through a free-flight experiment is the cross-coupling between the manipulator dynamics and the control systems of the servicer and target spacecraft.

In addition to the extensive component development which has been carried out in each of the core enabling technology areas, there have been several attempts to integrate them into a full system demonstration. The most ambitious was the Satellite Servicer System Flight Demonstration which was jointly sponsored by NASA and DoD. This program was to have drawn upon the hardware base described above as well as the Orbital Maneuvering Vehicle (OMV) and FTS programs to produce a free flying orbital demonstration of the three critical servicing technologies. Budgetary constraints at the sponsoring agencies led to its cancellation just as the contractors were about to begin design definition work.

Air Force Space Systems Division recently sponsored a concept study addressing rendezvous, docking, and refuelling on a smaller scale using an expendable vehicle. This study was carried out by NASA-Jet Propulsion Laboratory. Various means of carrying this concept forth to a flight demonstration are under consideration. One promising approach might be to use the SPAS hardware owned by SDIO which has already flown twice and will be reused for several additional experiment payloads.

We cannot ignore the fact that other nations have been active in this area, and the first flight demonstration may be performed by either the Japanese or Europeans. In particular, the Japanese ETS-7 spacecraft appears to have this as its goal. The ETS-7 will demonstrate the 3 key technologies described earlier.

8. IMPLEMENTATION OF SERVICING

One of the problems which must be addressed in order to obtain the benefits of on-orbit servicing is how this technology might be inserted into operational systems. There is a very long lead time in the development of new series of spacecraft and there is an understandable reluctance to make provisions for a capability which has not been fully integrated and demonstrated. Likewise, it is difficult to develop a support infrastructure in the absence of established users.

An effective way to work out of this chicken and egg impasse might be to introduce servicing capabilities on a contingency basis. For example, a grappling fixture could be incorporated into a spacecraft design at minimal cost and mass penalty. There may not be plans to use this in the original operational concept, but it might enable some corrective action to be taken later in the case of unanticipated events, as has occurred so often in the past.

Likewise, the capability for on-orbit refuelling could be provided at relatively low cost to the user. A fill valve is normally provided in spacecraft propulsion systems to allow the satellite to be fuelled shortly before launch. Making this valve compatible with robotic operations and allowing accessibility to it would enable on-orbit refuelling to take place at some future time, should the need arise and a servicing system be developed.

These contingency provisions can be justified by the extremely high ratio of the benefits which might be achieved, if they were ever to be needed, to the relatively modest costs of incorporating them as standard spacecraft design features.

Session A2: SENSING AND DISPLAY

Session Chair: Prof. Neil Duffie

N93-11954

AUTOGUIDANCE VIDEO SENSOR FOR DOCKING

by Michael L. Book(EB24),
Thomas C. Bryan(EB24),
Richard T. Howard(EB24),
and Richard W. Dabney(ED13)

Marshall Space Flight Center NASA
Huntsville, Alabama 35812

The Automated Rendezvous and Docking system is composed of two parts. The first part is the sensor which consists of a video camera ringed with two wavelengths of laser diode. The second part is a standard Remote Manipulator System(RMS) target used on the Orbiter that has been modified with three circular pieces of retro-reflective tape covered by optical filters which correspond to one of the wavelengths of laser diode. The sensor is on the chase vehicle and the target is on the target vehicle. The ARAD system works by pulsing one wavelength laser diodes and taking a picture. Then the second wavelength laser diodes are pulsed and a second picture is taken. One picture is subtracted from the other and the resultant picture is thresholded. All adjacent pixels above threshold are blobbed together(X and Y centroids calculated). All blob centroids are checked to recognize the target out of noise. Then the three target spots are windowed and tracked. The three target spot centroids are used to evaluate the roll, yaw, pitch, range, azimuth, and elevation. From that a guidance routine can guide the chase vehicle to dock with the target vehicle with the correct orientation.

BACKGROUND INTRODUCTION

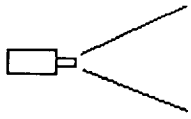
Past efforts in the rendezvous and docking area have been directed towards remotely piloted man-in-the-loop systems(OMV) or man directed systems (Orbiter). These video or visual based systems require a very large data stream to the ground operator or a man located on the rendezvous vehicle. To efficiently operate unmanned vehicles, minimize risk for hazardous or remote operations and reduce workload, automated rendezvous and docking (ARAD) techniques should be developed that will have application to the Cargo Transfer Vehicle(CTV), the Mars/Lunar Mission Vehicles and potentially the Shuttle Orbiter.

Recently, docking systems have been under investigation which use transponders and reflectors on the target vehicle and laser, radar and various computer based sensors have been used on the chaser vehicle for on-board range, range rate, and attitude determinations to support the ARAD function. Much work in this field has reached the level of real time system simulations and sensor testing at various NASA centers including the Marshall Space Flight Center(MSFC) and Johnson Space Flight Center(JSC). To define, develop, test, and evaluate various ARAD systems, a coordinated NASA effort is required. This effort will produce an implementation of an ARAD system that will have application to the Cargo Transfer Vehicle(CTV), Lunar/Mars Mission vehicles, and remote satellite servicing.

SYSTEM DESCRIPTION

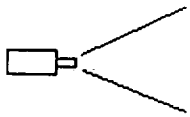
There are five major components in any docking autoguidance system. They are: the autopilot, the control system, the docking latches, the sensor, and the docking target. The autopilot generates vehicle commands from the autoguidance sensor outputs. The vehicle control system executes the commands through control system thrusters for a spacecraft or joint motors of a robot arm. The docking latches could be a three point docking mechanism or a more complex mechanism to lock the chase and target vehicles together. This report concentrates only on the autoguidance video sensor development using simple retro-reflectors for a target. This report is divided into two sections: the present system and new developments.

The video docking sensor concept consists of five main components: laser illuminators, retro-reflective targets, a video camera, a frame grabber board, and a microprocessor. The laser illuminators are wide-angle laser diodes at two different wavelengths, 780nm and 830nm. The diodes are arranged such that they cover a 30-by-30 degree field-of-view. The target is composed of three circular pieces of retro-reflective tape each covered by 830nm filters that are mounted on an RMS target in a line with the center reflector on the center pole and the two outer reflectors equally spaced from the pole. This configuration is much more sensitive for yaw and pitch at zero degrees than a four corner flat target would be. The target is only 14.5 inches wide and the pole is four inches high. The sensor takes a picture with the 830nm laser diodes on and then the sensor takes another picture with the 780nm laser diodes on. The second picture is subtracted from the first and then a threshold is subtracted from the resultant image to give the reflector positions along with some noise(see figure 1: Range, azimuth, and elevation can be converted to X, Y, and Z ranges).



830nm Laser Diodes

1st Digitized Picture

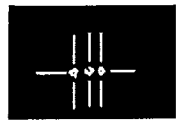


780nm Laser Diodes

2nd Digitized Picture

Image 2 subtracted from Image 1

Threshold subtracted from differential image



Centroids found for each spot

Tracking windows established

DISPLACEMENT
X–Y–Z
Roll–Pitch–Yaw

Relative positions and attitudes calculated

Position and attitude information sent to guidance algorithm

Figure 1: Steps to the Overall System Operation.

The software consists of two parts: initial target acquisition and reflector tracking.

The initial acquisition loop has three parts to it and they are: erosion and dilation, picking out bright spots, and identifying target reflectors out of noise spots.

Erosion and dilation are simply zeroing out the edge of all bright spots and then recreating the edge on all remaining bright spots. This elliminates all single point noise spots. Also, there is an edge created by pipes on target satellites which this elliminates along with any other edge effect. The target spots can not be too small or they will disappear.

There are two steps to picking out the bright spots and they are: grouping all adjacent pixels into spots, and calculating the X and Y centroids of each spot from the pixel data. The sensor scans the pixel data from left to right, top to bottom keeping track of the X coordinate(horizontal) and the Y coordinate(vertical) positions at all times. When a pixel is found with the intensity above the threshold, all adjacent pixels are checked to see if they are above the threshold. Then adjacencies to that layer of pixels are checked. All pixels above the threshold are zeroed out in the image as they are found so that they are not counted more than once. Adjacencies are checked for layer of pixels after layer of pixels until the spot is completely zeroed out of the image and stored with the X and Y coordinates. Then the scan continues where the first pixel for the spot was found. The data, X coordinates, and Y coordinates are then used to find the X and Y centroids of each spot.

$$X\_Centroid = X\_Factor / Data\_Factor$$

$$Y\_Centroid = Y\_Factor / Data\_Factor$$

$$X\_Factor = (Data\_1 * X\_Coordinate\_1) + (Data\_2 * X\_Coordinate\_2) + \ldots + (Data\_n * X\_Coordinate\_n)$$

$$Y\_Factor = (Data\_1 * Y\_Coordinate\_1) + (Data\_2 * Y\_Coordinate\_2) + \ldots + (Data\_n * Y\_Coordinate\_n)$$

$$Data\_Factor = Data\_1 + Data\_2 + \ldots Data\_n$$

After the X and Y centroids for all the spots are stored, the three target reflectors need to be identified from the noise spots. There has to be a minimum of three spots to even start to recognize the target. If there are three or more spots, these spot locations have to correspond to possible yaw and pitch target configurations to be identified as the reflectors. There is a triple nested loop for this section of code which examines three spots at any one time (any possible three spots is called a triplet). To reduce processing time, only non-redundant triplets are examined.

There are several steps inside the innermost loop needed to see if a triplet is the recognized reflector target. The first step is to calculate the length squared of each side of a triangle where the corners correspond to the triplet X and Y centroid locations. The second step is to find the base length or maximum length among the three lengths, which is the possible pixel length between the two outer reflectors. If there are two lengths of the same magnitude with one length shorter than the first two or if all three lengths are the same magnitude then there is no base length and the triplet is not the reflector target. The third step is to calculate the angle of the base length with respect to the horizontal(roll angle of possible reflector target). The fourth step is to use the roll angle to calculate an X and Y coordinate axis shift so that no matter how the target is rolled, accurate yaw and pitch target criteria can be used. These are the axis tranformation equations:

$$Roll = Atan\left(\frac{(First\_Y\_Centroid - Third\_Y\_Centroid)}{(Third\_X\_Centroid - First\_X\_Centroid)}\right)$$

Except 90 degrees. Assume centroid are sorted.

$$Center\_X = 0.5 * (Outer\_Left\_Spot\_X\_Centroid + Outer\_Right\_Spot\_X\_Centroid)$$

$$Center\_Y = 0.5 * (Outer\_Left\_Spot\_Y\_Centroid + Outer\_Right\_Spot\_Y\_Centroid)$$

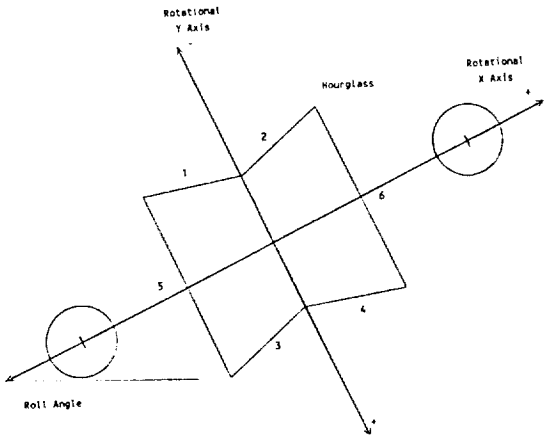
$$Translational\_X\_Shift = Center\_X\_Spot\_Centroid - Center\_X$$

$$Translational\_Y\_Shift = Center\_Y\_Spot\_Centroid - Center\_Y$$

$$Rotational\_X\_Shift = (Translational\_X\_Shift * Cos(Roll)) + (Translational\_Y\_Shift * Sin(Roll))$$

$$Rotational\_Y\_Shift = (Translational\_X\_Shift * Sin(Roll)) + (Translational\_Y\_Shift * Cos(Roll))$$

The reason for the axis tranformation is simple. There is a region centered between the two outer spots that is shaped like an hourglass with curved sides. The curved sides are approximated by four linear equations (see figure 2). This region corresponds to where the center spot would be if it were within 40 degees in yaw and pitch for the triplet to be the reflector target. The transformations are designed for the left-to-right increasing X coordinates and top-to-bottom increasing Y coordinates. Range is used as a target determining test for a reacquire. The final test is to compare the number of pixels in each spot to each other and if the numbers are close then accept the triplet as the target.



(1) Y = (0.238*X)-(Y_Intercept*Pixel_Length_Of_Target)
(2) Y = (-0.238*X)-(Y_Intercept*Pixel_Length_Of_Target)
(3) Y = (-0.238*X)+(Y_Intercept*Pixel_Length_Of_Target)
(4) Y = (0.238*X)+(Y_Intercept*Pixel_Length_Of_Target)
(5) X = -X_Intercept*Pixel_Length_Of_Target
(6) X = X_Intercept*Pixel_Length_Of_Target

X_Intercept = 0.22375 for small target
Y_Intercept = 0.17141 for small target

Figure 2: Hourglass Shaped Region Between Outer Spots.

Reflector tracking is done at two times per second using one frame grabber. First a frame is taken with the 830nm laser diodes active. Next, a frame is taken with the 780nm laser diodes active. The frame grabber subtracts the 780nm frame from the 830nm frame. Then the computer draws windows around the three reflectors and calculates the centroids of the windows. Range is used to calculate window size and window threshold(only pixel intensities above the threshold are included in the X and Y centroid calculations). The tracking itself (where to place next windows) is done by subtracting the the present X and Y centroids from the previous X and Y centroids to get the distances that are added to the present X and Y centroids to predict where the next windows should be. This works well for a constant rotational or translational velocity. At long range any chase vehicle yaw or pitch can shift the field-of-view enough for the reflectors to slip out of the windows because of the change in velocity. The same is true for close range translations. To counter this, translational and rotational accelerations should be sent from the control system Kalman filter back to the sensor. Then the X and Y centroids are used to calculate the roll, yaw, pitch, range, azimuth, and elevation (see appendix) which are sent to the control system.

The metnod to calculate the orientation information in the appendix is to first calculate roll using the equation from the acquisition section of this report. Then calculate Max_ Length between the two outer reflectors. Next, calculate the Rotational_(X and Y)_Shifts using equations from the acquisition section of this report. After that, use that information to calculate the yaw and pitch simultaneously. Finally, use the yaw to calculate range, azimuth, and elevation. Yaw, pitch, range, azimuth, and elevation all need to be calibrated for each lens used in any video auto-guidance system. Calibrated yaw should be used to calculate pitch, range, azimuth, and elevation. Calibrated range should be used to calculate azimuth and elevation.

PERFORMANCE/STATUS

The present system works fairly well within twelve meters of the docking target. With the new geometry algorithm(see appendix) the range accuracy has been calibrated to one percent or below. The yaw, pitch, azimuth, and elevation have been calibrated to below one degree in accuracy. Dynamically, the system performs docking maneuvers with a reasonable degree of reliability using the air-bearing vehicle on the MSFC flatfloor facility, bringing the three point docking mechanism into latch position smoothly. Continuing testing with the Dynamic Overhead Target Simulator duplicating the full dynamic motion of a free flyer(like CTV) showed good correlation of sensor outputs and realtime relative positions. The autoguidance video sensor will be used to measure and guide the Space Station Freedom common berthing mechanism test article during dynamic berthing tests and to control the latching sequence.

NEW DEVELOPMENTS/ENHANCEMENTS

The current sensor system has some limitations and its performance can be enhanced through some new hardware improvements, some of which are now commercially off the shelf. Other changes will require some development along with continued testing and integration with more of the full system.

To increase the range of the system, two targets may be used. The large target would be three corner cube reflectors or three circular pieces of microprism. Both of these reflector

types return a much larger signal to the camera and the larger the target the more the resolution at range. Corner cubes have been used at 40 meters away. Corner cubes have one problem and that is, a limited target yaw or pitch angle before the spot size is significantly reduced. Microprism may overcome this problem. The small target would be the present target used now for close range because a large target image would grow too large for the camera. Using a two target system could help determine roll ambiguity because the offset between the two targets is known. Otherwise one of the outer reflectors would have to be larger than the other to distinguish between the two.

Another target concept is the optical collector. The entire target could be made into an optical collector that brings all the light to a central point where optical fibers send the light back out to the three light output positions(where the reflectors were). All three points would have the same intensity no matter how close the target is to the camera. There are no glass filters to cause any starring (glass reflects both wavelengths at just the right incident angle causing spot loss in the differenced image). The intensity would be far brighter because all the light on the entire target is being sent back at three points instead of just the light hitting the reflectors which means that this target could be seen from a much farther range.

In order to significantly improve the speed of the sensor system to fifteen readouts per second, two frame grabbers that are pre-programmable are required. In such a system, the two frame grabbers would have the same program loaded in before the docking run by the 386-based computer. One frame grabber grabs a frame with the 830nm laser diodes active. Then it grabs a frame with the 780nm laser diodes active. Next, it subtracts the 780nm frame from the 830nm frame(only in the three reflector tracking windows). Finally, it centroids the windows, sends the centroids to the computer, and receives the next window locations from the computer. The other frame grabber follows the same program steps. While one frame grabber is grabbing two frames the other frame grabber is performing the subtraction, centroiding, and window updates. The computer takes the centroids and calculates the yaw, roll, pitch, range, azimuth, and elevation and sends this information to the guidance system. The computer also controls the laser diodes and calculates the windows to send to the frame grabbers.

Improving the camera optics can improve the performance of the system. One improvement would be to control the integration time of the camera CCD. In sunlight the CCD may be saturated in the brightest parts(noise spots and reflectors). In some instants the noise from sunlight shining on mylar may be much brighter than the reflectors. A threshold above the reflector brightness may be set up as well as a threshold below the reflector brightness. But if the camera is saturated, this information is lost.

Another way of improving the optics is a concept put forth by a company called TRW. In this concept there are two cameras one with an 830nm optical filter covering and one with a 780nm optical filter covering. At the entrance to the sensor there is a lens and a beam spliter. Ringing the lens are the output optical fibers. Each input to these fibers is a laser diode (830nm or 780nm). In this concept both wavelengths of laser diode are active at the same time. The returned signal goes to both cameras through the beam spliter at the same time(both cameras and frame grabber are synchronized to the same clock). The frame from the 780nm camera is subtracted from the frame from the 830nm camera (only in the reflector readout windows). The remaining steps are just like the present system. In this way image differences that translate into noise on the resultant image are completely absent. If a camera malfunctions there is a backup. Both cameras can have a filter with a wide enough bandwidth to allow both wavelengths through that can be snapped into place where the tighter filter was if there is a camera malfunction. Then this system would operate just like the present system. If this system were implemented along with the two pre-programmable frame grabbers then the resultant system could possibly produce a thirty readout per second data rate.

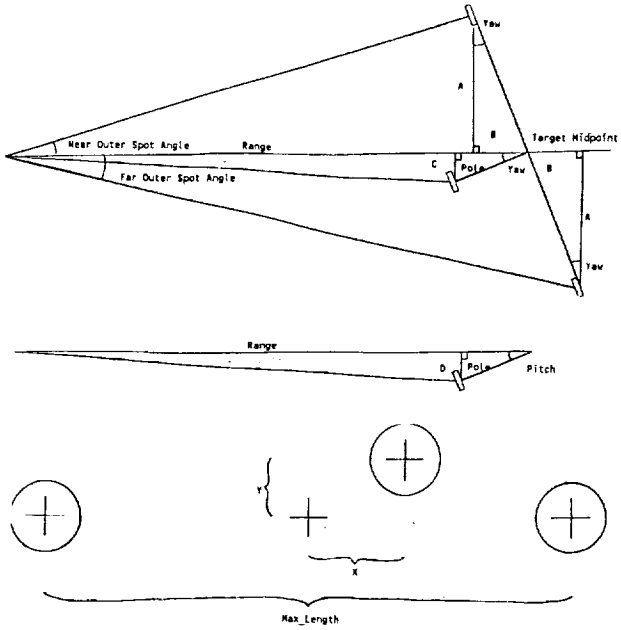
Another way to improve the sensor system is to have more than one camera each with a different lens and a different set of laser diodes. Each camera setup would be for a different range. At short range a wide field-of-view lens with laser diodes would be used. At longer range a narrow field-of-view lens with beam collimators to narrow the laser wide beam half power angle would be used with the laser diodes. The chase spacecraft would approach the target spacecraft with the longer range camera setups switching to the shorter range camera setups at the switchover ranges. This would greatly increase the range of the overall

system if implemented.

CONCLUSION

The current autoguidance video sensor, by acquiring and tracking three simple retroreflectors, provides accurate range and angular measurements for realtime docking of spacecraft. Any number of combinations of the above improvements may be implemented in the final system to create a highly reliable docking system with a reasonably good range from acquisition to the final docking. Other systems such as radar, laser rangers, GPS, etc will add autonomous rendezvous capability to the advanced docking system to provide fully Automated Rendezvous and Docking to support CTV, Mars/Moon missions and remote satellite servicing.

APPENDIX



A= 0.5*Target_Length*Cos(Yaw)
B= 0.5*Target_Length*Sin(Yaw)
C= Pole_Length*Sin(Yaw)*Cos(Pitch)
D= Pole_Length*Sin(Pitch)
2*A corresponds to Max_Length
A+C corresponds to 0.5*Max_Length+Abs(X)
so
((A+C)/(2*A))= (0.5*Max_Length+Abs(X))/Max_Length
S=(Pole_Length/Target_Length)
U=(Abs(X)/Max_Length)
0.5+ S*Tan(Yaw)*Cos(Pitch)= 0.5+U
U=(S*Tan(Yaw)*Cos(Pitch))
and D corresponds to Abs(Y)
so
(D/(2*A))=(Abs(Y)/Max_Length)
V=(Abs(Y)/Max_Length)
V= S*Sin(Pitch)/Cos(Yaw)

Sin(Pitch)=(1/S)*V*Cos(Yaw)
Cos(Pitch)=(1/S)*U/Tan(Yaw)
2 2 2
Sin (Pitch)=(1/S) *V *Cos (Yaw)
2 2 2
Cos (Pitch)=(1/S) *U/Tan (Yaw)
2 2
Sin (Pitch)+Cos (Pitch)= 1
2 2 2 2 2 2
(1/S)*(V *Cos (Yaw)+U *Cos (Yaw)/Sin (Yaw))= 1
2 2 2 2 2 2
U *Cos (Yaw)+V *Cos (Yaw)*Sin (Yaw)= S *Sin (Yaw)
2
Sin (Yaw)= 0.5*(1-Cos(2*Yaw))
2
Cos (Yaw)= 0.5*(1+Cos(2*Yaw))
2 2
0.5*U *(1+Cos(2*Yaw))+0.25*V *(1+Cos(2*Yaw))*(1-Cos(2*Yaw))=
2
0.5*S *(1-Cos(2*Yaw))
2 2 2 2 2
0.5*U +0.5*U *Cos(2*Yaw)+0.25*V -0.25*V *Cos (2*Yaw)=
2 2
0.5*S -0.5*S *Cos(2*Yaw)
2 2 2 2 2 2 2
0.25*V *Cos (2*Yaw)-0.5(U +S)*Cos(2*Yaw)+(0.5*S -0.5*U -0.25*V)= 0
2
G= 0.25*V
2 2
H= -0.5*(U +S)
2 2 2
I=(0.5*S -0.5*U -0.25*V)
2 0.5
Cos(2*Yaw)=(-H-(H -4*G*I))/(2*G)
Calibrated_Yaw=C1*(Yaw)+C2
where C1 is a slope and C2 is an angle offset (both determined by experimental data and linear regression)
Input Calibrated_Yaw into V= S*Sin(Pitch)/Cos(Calibrated_Yaw) to get Pitch.
Calibrated_Pitch=C3*(Pitch)+C4
where C3 is a slope and C4 is an angle offset (both determined by experimental data and linear regression)
for Y=0 (Pitch=0)
U= S*Tan(Yaw) then calibrate Yaw
for X=0 (Yaw=0)
V= S*Sin(Pitch) then calibrate Pitch

Near_Outer_Spot_Angle= Atan(A/(Range-B))
Far_Outer_Spot_Angle= Atan(A/(Range+B))
Target_Width_Angle= Near_Outer_Spot_Angle+Far_Outer_Spot_Angle
Target_Width_Angle= Atan(A/(Range-B))+Atan(A/(Range+B))
Tan(Target_Width_Angle)= (Tan(Near_Outer_Spot_Angle)+Tan(Far_Outer_Spot_Angle)) / (1-Tan(Near_Outer_Spot_Angle)*Tan(Far_Outer_Spot_Angle))
Tan(Target_Width_Angle)= ((A/(Range-B))+(A/(Range+B))) / (1-(A/(Range-B))*(A/(Range+B)))
2 2 2
Tan(Target_Width_Angle)= 2*A*Range/(Range -A -B)
2 2 2
Range -(2*A/Tan(Target_Width_Angle))*Range-(A +B)= 0
-3
Target_Width_Angle= 1.5338*10 *Max_Length
H= -(2*A/Tan(Target_Width_Angle))
2 2
N= -(A +B)
2 0.5
Range= 0.5*(-H+(H -4*N))
Calibrated_Range=Range+C5
where C5 is the front focal length of the lens plus any other added length
Pix_Shift=651.975*Atan(A/(Calibrated_Range-B)) to shift image center to true target center
if Yaw greater than or equal to 0 (651.975 is pixels per radian)
XA=(Pix_Shift/Max_Length)-0.5
If Yaw<0
XA= 0.5-(Pix_Shift/Max_Length)
X_Shift=XA*Cos(Roll)
Y_Shift= -XA*Sin(Roll)
Center_X= X_shift+Center_X
Center_Y= Y_shift+Center_Y
-3
Azimuth=1.5338*10 *(Center_X-255.5)
-3
Elevation=1.5338*10 *(191.6-Center_Y)
Calibrated_Azimuth=C6*(Azimuth)+C7
where C6 is a slope and C7 is an angle offset (both determined by experimental data and linear regression)
Calibrated_Elevation=C8*(Elevation)+C9
where C8 is a slope and C9 is an angle offset (both determined by experimental data and linear regression)

Model-Based Vision for Space Applications

Karen Chaconas
Marilyn Nashman
Ronald Lumia

National Institute of Standards and Technology
Robot Systems Division
Gaithersburg, MD 20899

Abstract

This paper describes a method for tracking moving image features by combining spatial and temporal edge information with model based feature information. The algorithm updates the two-dimensional position of object features by correlating predicted model features with current image data. The results of the correlation process are used to compute an updated model. The algorithm makes use of a high temporal sampling rate with respect to spatial changes of the image features and operates in a real-time multi-processing environment. Preliminary results demonstrate successful tracking for image feature velocities between 1.1 and 4.5 pixels every image frame. This work has applications for docking, assembly, retrieval of floating objects and a host of other space-related tasks.

1. Introduction

The ability to visually track an object during arbitrary motion is an important part of interacting with the environment. Humans adeptly recover three-dimensional structure of an object from its rigid-body motion in order to accomplish manipulation, locomotion, and object recognition [6]. Motion and edge information are known to be important cues to the recovery of object structure. Understanding the relative motion between an object and an observer aids not only in the recovery of object structure but also provides useful information required to interact with the objects. In order to visually track an object, the object's orientation and position must be rapidly updated. In six degree-of-freedom robotic applications, real-time camera images can provide a dense stream of data from which to extract object features and recover rigid body motion.

An object's three-dimensional structure can be reconstructed from the projection of its motion onto a sequence of two-dimensional images [22]. Two methods for measuring visual motion are detection of spatio-temporal intensity changes and feature tracking [18]. The measurement of spatio-temporal intensity changes can be accomplished using correlation models, energy filters [1] [4], or gradient techniques [2] [13].These algorithms provide a description of two-dimensional image motion as a result of changes in intensity in the image. They cannot differentiate between intensity differences due to changes in viewer motion and intensity changes produced by object motion with respect to a light source. Thus, they are not suited to the task of tracking a moving object. The class of algorithms that measure visual motion by tracking features provides a means to directly measure physical motion in the world. These algorithms, however, have the disadvantage of requiring feature correspondence between images.

By combining the feature tracking approach with model-driven techniques, the feature tracking process is constrained and the correlation of features between frames is simplified. This results in a computationally inexpensive and accurate system. This paper describes an approach designed to achieve these goals and the implementation of this approach in the Intelligent Controls Group (ICG) laboratory at the National Institute of Standards and Technology. The next section discusses model-based feature tracking methods and, in particular, the two-dimensional tracking method we use. Section 3 details the implementation of the algorithm in our lab. The fourth section quantifies the accuracy and speed of this algorithm in preliminary experiments by tracking a planar target, and the final section discusses the implications of these results.

2. Model-based Feature Tracking

Model-based feature tracking correlates image features with object model features to take advantage of model information. Correlation between extracted features and an object model can be performed in either a two-dimensional [9] or three-dimensional [12] [19] frame-of-reference. A useful survey of work done using these methods can be found in [20]. Three-dimensional tracking involves comparing a set of two dimensional features extracted from an image to a three-dimensional model. In the most general case, this means solving the three-dimensional recognition problem for each successive image frame. The three-dimensional position and orientation of the feature are computed by analyzing images taken at different positions. Correspondence of features between image frames is determined, and the three-dimensional feature position is computed and matched to the model. This process is time consuming computationally expensive.

A more efficient way of using a three-dimensional frame of reference for model matching involves computing only the changes in structure position and orientation between image

frames [20]. The set of extracted two-dimensional features that must be matched with a three-dimensional model is thus reduced. By projecting the model into image frame coordinates, the search space is further reduced since the approximate location of each model feature in the two-dimensional image frame is known. The information extracted and used to update the model is usually quite accurate, and since all surfaces of the object model are available, problems of changing viewpoints or occlusion are handled. However, the computational complexity of these algorithms prevent their use in real-time applications.

The model-based feature tracking approach where the matching occurs in two-dimensions is a less complex alternative to the tracking problem. It assumes the ability to process a continuous sequence of two-dimensional images in real-time [9]. Zero or one-dimensional features such as vertices, centroids, or edge segments are extracted and matched to a two-dimensional projection of the model. Analogous to the three-dimensional case, the process is simplified by computing motion features which represent the changes in position and orientation between image frames. Since the temporal sampling rate is high, there is little change in position and orientation between successive frames and the correlation between observation and model is simplified. The two-dimensional model is continually updated based upon the most recent observation. Tracking in two dimensions continues exclusive of additional information as long as the object motion is continuous. A model update obtained from three-dimensional information is required if there is occlusion or a change of direction causing loss of two-dimensional information. In general, two-dimensional feature tracking is an inexpensive method of correlating observations with model information and is well-suited for real-time applications [20].

The approach in our lab is based on model-based feature tracking in two dimensions. There are two phases to the method that are used: initialization and tracking. During initialization, an operator defines the position of the object features in the image at the point where tracking is to begin. In our preliminary experiment, a planar object is attached to a pendulum that is constrained to two-dimensional motion and tracked while it is swinging. A sequence of camera images that demonstrates this motion is digitized and stored. This sequence is used as input to our algorithm and provides us with repeatable motion for our experiments. Since the path is repeatable, the operator can select the feature points, which are the object corners, from any one of the images in the sequence. Tracking begins when all features selected by the operator are within an acceptable distance from the extracted image corners. Successful tracking continues while the extracted image features remain within a predefined distance from the predicted corners. Tracking is lost when this distance is exceeded.

Figure 1 depicts an overview of the algorithm during the tracking phase. In this figure, $I(x,y)_t$ refers to the intensity function at a pixel located at position (x,y) at time t. Motion and edge features from a sequence of images are correlated with model information. Model-based tracking involves segmentation and correlation of observed data with model data and the prediction of the model position at the next time interval.

During the segmentation phase, optical flow and edge orientation are extracted from an incoming sequence of images. This information represents the temporal and spatial edge information respectively. The image flow results from changes in intensity between frames and a temporal differencing algorithm is used to measure these changes. Incoming images are smoothed using a Gaussian convolution, G*, to diminish the effects of spurious noise in the image. Two temporally-consecutive, smoothed images are subtracted from each other in order to detect any change in intensity due to motion between the frames (Equation 1).

$$\frac{\partial}{\partial t} I(x,y) = G^*(x,y)_t - G^*(x,y)_{t+1} \qquad [1]$$

All non-moving features in the image disappear in this difference image since the grayscale value of a pixel in the second frame is being subtracted from the identical grayscale value in the first frame. The resulting optical flow image is thresholded to produce a binary image. This operation results in a segmented scene reflecting changing intensity values between successive images. However, motion segmentation occurs whether the changing intensity is due to relative motion between the camera and the object or between the object and a light source and can therefore be an ambiguous basis for segmentation.

By requiring object features in an image to adhere to consistent spatial as well as temporal properties, the segmentation of features is more robust. Edges are also extracted during the segmentation process to provide additional information. Spatial orientations of edge points are computed directly from the sequence of input images. This information is used to determine the spatial orientations of the image motion points. Since the positions of the edge points on the object change between frames, a dense set of edge points is extracted from the sum of two temporally-consecutive, smoothed images. A two-dimensional spatial gradient operator is applied to all points (x,y) in the image. The actual direction, θ, of each point in the image is defined to be perpendicular to the direction of the gradient of the intensity function f(x,y) at that point:

$$\theta = \operatorname{atan}\frac{(\nabla_y f)(x,y)}{(\nabla_x f)(x,y)} + \frac{\pi}{2} \qquad [2]$$

Since the edge extraction and the motion extraction operations are performed in parallel on the same input images, this provides the advantage of having edge orientation information for most motion pixels.

The next step, correlation of the extracted motion points with the model edges, makes use of the segmentation information. Each motion pixel is either labelled or discarded depending on its similarity to the model. The labelling process is based on two criteria. The first criterion is the two-dimensional spatial proximity of a motion point to the model lines.The second criterion is the similarity of direction of the edge at the location of this motion with the angular direction of the model line.

The description of each line in the model includes the slope-intercept form of the line, the coordinates of the endpoints of each line segment, and the angular direction of the line. The first step in the correlation process compares the angular direction of the model line with the edge direction of the candidate motion point to determine if the angular disparity is within an acceptable range:

$$|\theta_{model} - \theta_{data}| < \delta \qquad [3]$$

If this condition is satisfied, the distance d is computed between the point at image coordinate (x_i, y_i) and each of k lines used to define the object model using equation [4]:

The minimum distance between the image motion point and each of k model lines is used to determine if that point is less than a distance threshold, ζ, from the line. The point is labelled as belonging to the model line segment it best matches when both the spatial proximity and orientation conditions are satisfied.

$$d = \frac{A_k x_i + B_k y_i + C_k}{(A_k^2 + B_k^2)^{1/2}} \qquad [4]$$

where $(A_k x + B_k y + C_k)$ is the general form for the k^{th} line in the model.

After the motion points are segmented, a line-fitting technique is used to update the two-dimensional position of each model line. The line-fitting technique uses a least-squares linear regression which minimizes the squared error in either the x or y direction. The equations that compute slope, m, and the y intercept, b, of the best fitting line through the n points (x_i, y_i) when minimizing the x direction error are:

$$m = \frac{n(\sum_i (x_i y_i)) - (\sum_i x_i)(\sum_i y_i)}{n\sum_i (x_i^2) - (\sum_i x_i)^2} \qquad [5]$$

$$b = \frac{(\sum_i y_i)(\sum_i x_i^2) - (\sum_i x_i)\sum_i x_i y_i}{n(\sum_i x_i^2) - (\sum_i x_i)^2} \qquad [6]$$

Minimizing the least square error in the x direction presents a problem as the line being fit approaches vertical. As this happens, the denominator in equation [5] approaches zero, and the fit becomes less accurate. A more accurate fit takes this into account and minimizes errors in both x and y. Comparison of the standard deviation of the x coordinates to that of the y coordinates gives a measure as to whether the line is more horizontal or more vertical. A larger standard deviation in x means the line tends toward the horizontal. When the standard deviation of the y coordinates is larger, a linear regression of x on y is used since the line is more vertical. In this case, the equation for the slope and intercept of the fitted line is given by

$$m = \frac{n(\sum_i y_i^2) - \sum_i (y_i)^2}{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)} \qquad [7]$$

$$b = \frac{-((\sum_i x_i)(\sum_i y_i^2) - (\sum_i y_i)(\sum_i x_i y_i))}{(n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i))} \qquad [8]$$

If the standard deviation of x coordinates is less than a predefined threshold value, the line is considered to be vertical, and therefore the slope is undefined.

The computed lines are intersected to determine the two-dimensional corner positions of the object model. Each corner point (x_c, y_c) is computed by solving for the intersection of the two fitted lines which contain the corner as an endpoint. The distance between the computed corners and the model corners is used to determine the proximity of the results with the prediction. If the resulting distance is within an acceptable limit, ε, then a successful match has been detected (Equation [9]):

$$\sqrt{(x_c - x_{model\,k})^2 + (y_c - y_{model\,k})^2} < \varepsilon \qquad [9]$$

When the distance exceeds ε, tracking is lost; distances less than ε indicate tracking. When tracking is successful, the corners are filtered using an exponential smoothing filter [17] to predict the corner positions at the next time interval. Each corner is filtered independently of the others since the object motion isn't necessarily parallel to the image plane and the corners will move at different rates in the image plane. The filtering of each corner is done using a weighted average of the current and all past positions of that corner with exponentially decreasing weights (Equation 10).

$$(x, y)'_t = \alpha (x, y)_t + (1 - \alpha)(x, y)'_t \qquad [10]$$

In this equation, $(x, y)'_t$ is the filtered corner position and $(x, y)_t$ is the unfiltered corner position. The value of $(x, y)'_t$ before any previous iterations is set to the position of the corner when tracking is initially begun. The smoothing constant, α, is in the range $0 < \alpha \leq 1$ and, in our case, is chosen to be 0.2. The corner is filtered again using equation 11.

$$(x, y)''_t = \alpha (x, y)'_t + (1 - \alpha)(x, y)''_t \qquad [11]$$

The value of $(x, y)''_t$ before any previous iterations is also set to the position of the corner when tracking is initially begun. After the filtered values are determined, the predicted corner position $(x,y)_{t+1}$ is computed using equation [12].

$$(x, y)_{t+1} = (2 + \frac{\alpha}{1 - \alpha})(x, y)'_t$$

$$(-(1+\frac{\alpha}{1-\alpha})\,(x,y)_t^{\tau}) \quad [12]$$

The predicted model information is used to segment the extracted image flow information at time t+1. Predictions are computed each execution cycle regardless of whether updated observed corners are available. This allows predicted positions to be sent to the correlation process continually, though the spacing between successive positions decreases while no new data are being observed. Figure 2 shows the relative frequency with which predicted data are generated as compared to data extracted from incoming images over a period of about 1 second. Since the results of the tracking algorithm are used as feedback in a real-time system, the implementation of this algorithm must be fast enough to be stable. Section 3 describes the approach we use.

3. Implementation

Processing in the integrated vision testbed in the ICG laboratory is accomplished using a programmable real-time image processor, the Pipelined Image Processing Engine (PIPE)[1] [5] and a multiprocessor system as shown in Figure 3. In this figure, the large grey rectangles represent how the software processes are distributed on this hardware. The incoming images from a CCD camera are digitized by PIPE to provide 8-bit grayscale images that are 242x256 pixels in size. The images are processed by lookup tables, neighborhood operators, and arithmetic logic units that are defined in the PIPE application program. Smoothing, temporal integration, and edge and motion detection are performed on the grayscale images as described in equations [1] and [2]. The Iconic-to-Symbolic Mapper (ISMAP) stage of PIPE converts information from an image format to a symbolic list and is used to store the binary motion image as a list of pixel positions. In addition, the corresponding edge direction values are stored in the ISMAP iconic buffer where they are mapped onto the memory of one of the microprocessors via a specialized PIPE-VME interface board. Figure 3 displays these pipelined processes as black parallelograms.

The remaining software processes operate in real-time in the multiprocessing environment. They are implemented within the hierarchical sensory-interactive robot control system in our lab [7] [10] [15] [16] [21] that is defined in the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) [3]. The control system is composed of three parallel systems that cooperate to perform telerobot control (Figure 4). The task decomposition system breaks down objectives into simpler subtasks to control physical devices. The world model supplies information and analyzes data using support modules. It also maintains an internal model of the state of the environment in the global data system. The sensory processing system monitors and analyzes sensory information from multiple sources in order to recognize objects, detect events and filter and integrate information. The world model uses this information to maintain the system's best estimate of the past, current, and possible future states of the world. The processes are labeled according to their functional role in the NASREM architecture as sensory processing (SP), world modeling (WM), or task decomposition (TD) modules. Each device or sensor of the telerobot has a support process in each of the three columns of the control system. For example, the task decomposition functions associated with planning the actions for processing camera data reside in the task decomposition hierarchy; the world modeling functions for supporting those plans reside in the world model hierarchy, and the image processing techniques required for executing those plans reside in the sensory processing hierarchy. The world modeling support modules communicate asynchronously with the task decomposition and sensory processing systems. Data flows bidirectionally between adjacent levels within any given hierarchy. The interfaces to the sensory processing system allow it to operate in a combination of bottom-up (data driven) and a top-down (model driven) modes. Bottom-up processing involves the extraction of knowledge from sensory data, and top-down processing is used to correlate predicted information from the world model with extracted information from the environment. The interfaces between the sensory processing system and the world model allow updated information to be sent to the world model and predicted information or sensory processing parameters to be sent to the sensory processing system.

The implementation is based on the concept of cyclically executing modules which serve as the computational units for the NASREM architecture [11]. After initialization, all computations are performed by cyclically executing processes that communicate via global read-write interfaces. Each unit acts as a process which reads inputs, performs computations, and then writes output. Such a process always reads and executes on the most current data; it does not wait for new data to arrive since reliable cyclic execution requires that a module be able to read or write data with minimal delay. Reading and writing involve the transfer of data between local buffers and buffers in global memory. System software has been written to prevent data corruption during these transfers.

Three cyclically-executing software processes execute the model-based feature tracking algorithm. These reside on two of the three microprocessor boards. The remaining software process performs communication with PIPE. The PIPE communication process is a cyclically executing process which polls PIPE status, reads the ISMAP output produced by PIPE, and writes it to the appropriate common memory locations that it shares with the segmentation process. Though the amount of data transferred is large, on the order of hundreds to thousands of pixels every cycle of 60 Hz, the direct memory accessing provides a high rate of accessibility to the symbolic data. The execution time for this process takes an average of 90 ms on a 68020 processor for about 1400 motion pixels, an average sample for our tests described in the following sections. The correlation steps described in equations [3] - [4] and the line-fitting described in equations [5] - [8] are computationally intensive. The processing times for these operations depend on the number of motion points. This process requires

1. Commercial equipment and materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily the best for the purpose.

high bandwidth, and for this reason, the correlation and line-fitting process executes on a dedicated 68030 microprocessor at an average rate of 110 ms per execution cycle. Since the execution time is greater than that of the PIPE communications process, it always has new data at the beginning of its execution cycle. The resulting lines are written to a common memory buffer shared with the process which computes the corners of the observed object. This process computes all of the object corners in 2.1 ms and then updates the buffer shared with the filtering and prediction process. The filtering and prediction in equations [10] - [12] are used to obtain all of the corners in the predicted model, and this process executes in 3.1 ms. These processes are combined on one board, a 68020 processor since their total execution time is small compared to the other processes.

4. Results

A preliminary set of experiments to determine how accurately this model-based feature tracking algorithm can track an object was run for the case of simple translational velocity. In these experiments, a planar, rectangular object was mounted on a pendulum as shown in Figure 5. The pendulum is released at different heights to provide image motion at differing velocities. The only a priori knowledge about the object is the image positions of the four corners at an arbitrary point during the path of the pendulum that are used as a starting point for tracking as described in section 2. These points establish a crude model and are necessary to establish when the object has initially been matched. Once the observed data matches the initial model, the object is tracked by the single-camera vision system in the manner previously described.

In order to quantify the accuracy with which the model predicts the position of the observed data, the distance between the model corners and the actual corners is computed. This difference is used to determine the accuracy of the fit between the predicted and the observed data at varying object velocities. The accuracy of the tracking algorithm is also affected by the threshold parameters used. The threshold used in equation [4] controls the labelling of motion pixels and determines how closely the data can be correlated to the model. The threshold used to match the model corners to the observed corners in equation [9] determines how closely the model must match the observed data before tracking is lost. Our experiments consisted of three cases of varying object velocity. Velocity is measured in the image plane as the distance that an object feature moves between camera frames. The velocities tested are 1.1 pixels per frame, 3.9 pixels per frame, and 4.5 pixels per frame. For each velocity, the correlation threshold is tested at four values, 3 pixels, 5 pixels, 10 pixels and 15 pixels. In addition, the corner matching threshold is tested at 32 pixels, 26 pixels and 20 pixels. In all, twelve sets of data were collected for each velocity. Each set of data consists of 200 error measurements.

The threshold parameters were varied for three different object velocities measured in the image plane. Table 1 summarizes the results of experiments for the three velocities at varying correlation thresholds. The corner threshold remains constant at 20 pixels. Tables 2 and 3 are similarly organized except that the corner thresholds are set to 26 pixels and 32 pixels, respectively. Continuous tracking was performed successfully for all cases over a period of 200 iterations. By comparing the tables, it can be seen that the threshold used to determine successful tracking described in equation [9] does not play a significant role. Table 1 shows that for each object velocity, the mean error increases as the correlation threshold increases. This is a result of the fact that as the distance threshold is relaxed, there is a greater chance of misclassifying motion pixels. This effect is noticeable at faster velocities since there are more motion pixels available for processing. Also it can be seen that at distance thresholds of 3 and 5 pixels, the tracking error decreases with increasing velocity. This can be attributed to the value chosen for the smoothing constant α described in equations [10] - [12] which provides predictions more closely matching the observed data at a velocity of 4.5 pixels per 60 Hz. It is not clear that this trend would continue for higher velocities using the same smoothing constant. At distance thresholds of 10 and 15 pixels the tracking error increases as the velocity increases. This is caused by a greater number of motion pixels being present at higher velocities compounding the effect of the relaxed threshold. Similar conclusions can be drawn by analyzing Tables 2 and 3.

5. Conclusions

The method described in this paper successfully tracks moving image features by correlating a combination of extracted spatial and temporal edge information with an object model. The use of spatial information in the form of edge point orientations constrains the correlation process since motion points whose orientation is outside the orientation tolerance are discarded. This provides the advantage of being able to track an object in an unconstrained environment. Since a feature point has to be moving and in the correct orientation and position to be matched to the object model, other features can be in the field of view without being considered as part of the object. Another advantage of this algorithm is that the predicted model information can vary from the extracted image features up to 15 pixels and still track successfully. This is an improvement over algorithms that base correlation only on local properties. Preliminary results demonstrate successful tracking for image feature velocities between 1.1 and 4.5 pixels between image frames.

In the future, we plan to expand the modelling capability of our system to handle the appearance and disappearance of object features. By using a three-dimensional model we will be able to predict the most stable set of features to track for a given object pose. Modelling the motion of the object will enable us to provide pose predictions in the absence of sensory data. We also plan to continue the experiments on model-based feature tracking and to extend the scope of our algorithms to include processing on a stereo set of cameras [8]. Knowledge of the two-dimensional position of the same feature as viewed from two cameras will enable us to determine the position and orientation of the object in world-space using range from triangulation. This capability will allow us to supply feedback information to a manipulator system to aid in tasks involving tracking or grasping a moving part.

6. References

[1] Adelson, E. H., J. R. Bergen, "Spatio-temporal Energy Models for the Perception of Motion," *Journal of the Optical society of America A*, Vol. 2, No. 2, February, 1985, pp. 284-299.

[2] Albus, James S., Tsai-Hong Hong, "Motion, Depth, and Image Flow," *IEEE Robotics and Automation Conference*, Cincinnati, OH, May 13-18, 1990.

[3] Albus, J. S., H. G. McCain, R. Lumia.,"NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," NIST Technical Note 1235, Gaithersburg, MD, July, 1987.

[4] Allen, Peter K., "Real-time Motion Tracking Using Spatio-Temporal Filters," *Proceedings of the DARPA Image Understanding Workshop*, Palo Alto, TX, May 23-26, 1989.

[5] Aspex, Inc., "PIPE--An Introduction to the PIPE System," New York, 1987.

[6] Bolles, Robert C., H. Harlyn Baker, David H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Vol. 1, 1987, pp. 7-55.

[7] Chaconas, K., M. Nashman, "Visual Perception Processing in a Hierarchical Control System", NIST Technical Note 1260, Gaithersburg, MD, March, 1989.

[8] Chaconas, K., "Range from Triangulation Using An Inverse Perspective Method to Determine Relative Camera Pose," NIST Internal Report 4385, Gaithersburg, MD, August, 1990.

[9] Crowley, James L., Patrick Stelmaszyk, Christopher Discours, "Measuring Image Flow by Tracking Edge-Lines," *Proceedings of the 2nd International Conference on Computer Vision*, 1988, pp. 658-664.

[10] Fiala, J., "Manipulator Servo Level Task Decomposition," NIST Technical Note 1255, NIST, Gaithersburg, MD, October, 1988.

[11] Fiala, J. "Note on NASREM Implementation," NIST Internal Report 89-4215, Gaithersburg, MD, December, 1989.

[12] Gennery, Donald B., "Tracking Known Three-Dimensional Objects," *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA, August 18-20, 1982, pp. 13-17.

[13] Horn, B. K. P., B. Schunk, "Determining Optical Flow," *Artificial Intelligence*, Vol. 17, 1983, pp. 185-203.

[14] Jenkin, M., J. K. Tsotsos, "Applying Temporal Constraints to the Dynamic Stereo Problem," *CVGIP*, 33, 1986, pp. 16-32.

[15] Kelmar, L. "Manipulator Servo Level World Modeling," NIST Technical Note 1258, NIST, Gaithersburg, MD, March, 1989.

[16] Lumia, R., Fiala, J., Wavering, A., "The NASREM Robot Control System and Testbed," 2nd Intl. Symp. on Robotics & Automated Manufacturing, Albuquerque, NM, November, 1988.

[17] Montgomery, D. C., L. A. Johnson, and J. S. Gardiner, "Forecasting & Time Series Analysis," Second Edition, McGraw-Hill, New York, 1990.

[18] Spetsakis, Minas E., John Aloimonos, "Closed Form Solution to the Structure from Motion Problem from Line Correspondences," *Proceedings of the National Conference on Artificial Intelligence*, 1987, pp 738-743.

[19] Thompson, D. W., J. L. Mundy, "Model-based Motion Analysis - Motion from Motion," *Robotics Research: The Fourth International Symposium*, R. C. Bolles and B. Roth, eds., The MIT Press, Cambridge, MA, 1988, pp. 229 - 235.

[20] Verghese, Gilbert, Charles R. Dyer, "Real-time Model-Based Tracking of Three-Dimensional Objects," Computer Sciences Technical Report #806, University of Wisconsin - Madison, November, 1988.

[21] Wavering, A., "Manipulator Primitive Level Task Decomposition," NIST Technical Note 1256, NIST, Gaithersburg, MD, October, 1988.

[22] Waxman, Allen M., Kwangyoen Wohn, "Image Flow Theory: A Framework for 3-D Inference from Time-Varying Imagery," LSR-TR-1, Boston University, January, 1986.

[23] Waxman, Allen M., Jian Wu, F. Bergholm, "Convected Activation Profiles: Receptive Fields for Real-Time Measurement of Short-Range Visual Motion," *International Conference on Computer Vision*, April, 1988.

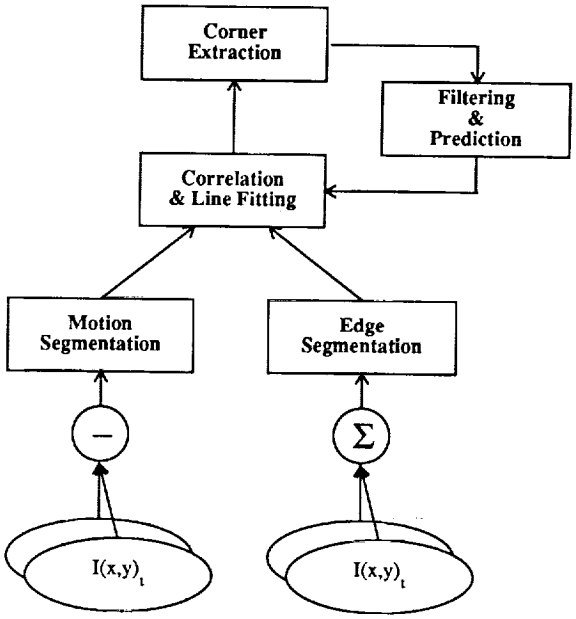


Figure 1. Model-based Feature Tracking Algorithm

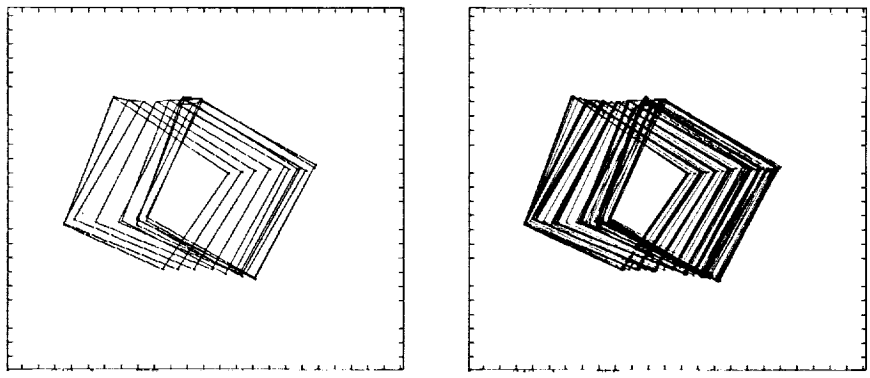


Figure 2. (a) Observed Object Positions (b) Predicted Object Positions

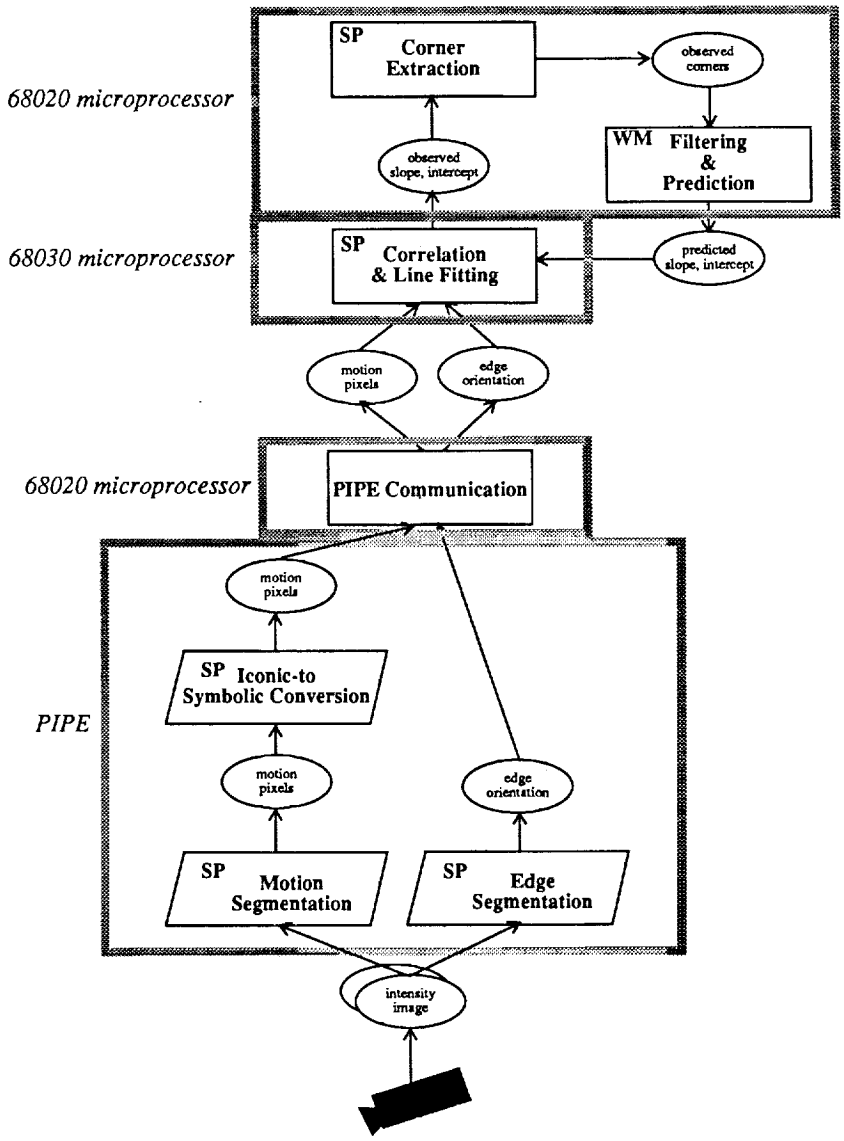


Figure 3. Implementation of Model-Based Feature Tracking Algorithm

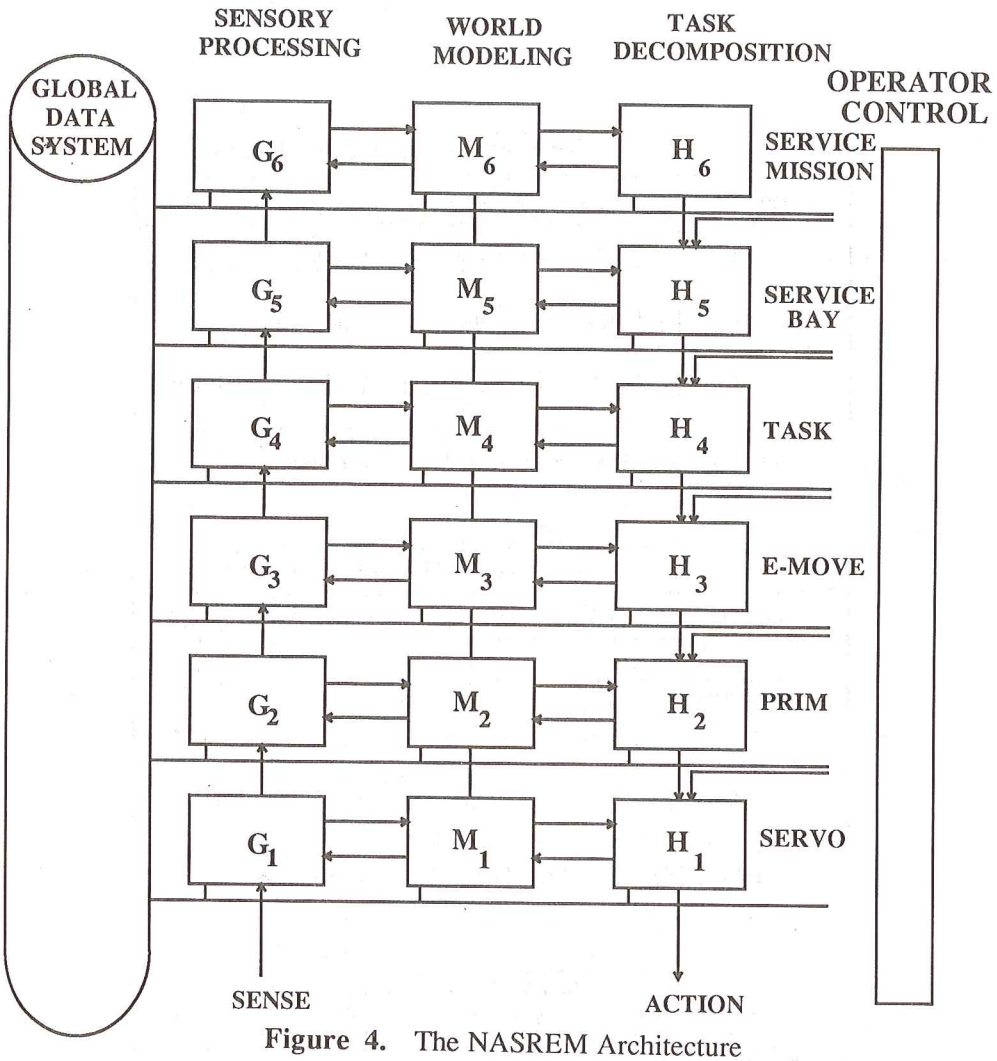


Figure 4. The NASREM Architecture

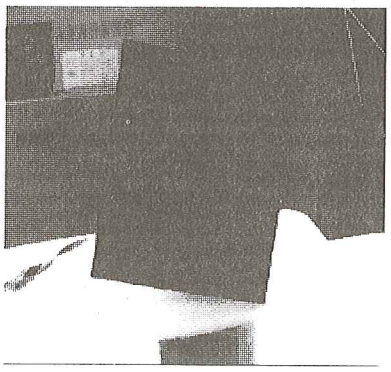


Figure 5. Experimental Scenario

	Distance Threshold for Correlation			
Velocity	**ζ = 3.0**	**ζ = 5.0**	**ζ = 10.0**	**ζ = 15.0**
1.1	0.302	0.338	0.335	0.407
3.9	0.045	0.096	0.096	1.151
4.5	0.009	0.024	1.174	1.368

Table 1. Mean Data Error Using Corner Threshold $\varepsilon = 20$

	Distance Threshold for Correlation			
Velocity	**ζ = 3.0**	**ζ = 5.0**	**ζ = 10.0**	**ζ = 15.0**
1.1	0.301	0.313	0.357	0.407
3.9	0.110	0.108	0.052	0.052
4.5	0.009	0.005	1.177	1.368

Table 2. Mean Data Error Using Corner Threshold $\varepsilon = 26$

	Distance Threshold for Correlation			
Velocity	**ζ = 3.0**	**ζ = 5.0**	**ζ = 10.0**	**ζ = 15.0**
1.1	0.302	0.313	0.348	0.407
3.9	0.031	0.109	0.097	1.160
4.5	0.001	0.003	1.174	1.273

Table 3. Mean Data Error Using Corner Threshold $\varepsilon = 32$

CONTACT DETECTION AND CONTACT MOTION FOR ERROR RECOVERY IN THE PRESENCE OF UNCERTAINTIES

Jing Xiao
Computer Science Department
University of North Carolina at Charlotte
Charlotte, NC 28223
E-mail: xiao@unccvax.uncc.edu

Abstract

Due to various kinds of uncertainties, a robot motion may fail and result in some unintended contact between the object held by the robot and the environment, which greatly hampers robotics applications on tasks with high-precision requirement, such as assembly tasks. Aiming at automatically recovering a robotic task from such a failure, this paper discusses, in the presence of uncertainties, contact detection based on contact motion for recovery. It presents a framework for on-line recognizing contacts using multiple sensor modalities in the presence of sensing uncertainties and means for ensuring successful compliant motions in the presence of sensing and control uncertainties.

1 Introduction

The issue of detecting and recovering errors of robot actions due to uncertainties (e.g., mechanical, control, modeling, manufacturing, and sensing uncertainties) is crucial for robotics applications on tasks with high-precision requirement, such as assembly tasks. Since errors of a robot action almost always lead to some unintended collisions between the object moved by the robot and some other objects, online recognition of those collisions or contacts is extremely important to recovery strategies. On the other hand, recovery motions are usually preferred to be contact motions, i.e., compliant motions, in order to reduce the effect of uncertainties via the physical constraints among objects.

The contact detection problem not only requires sensing and sensor-based reasoning but also demands them in greater precision with sensing uncertainties being taken into account. Fig. 1 shows an example to illustrate this. A robot is used to perform the peg-in-hole task as depicted in Fig. 1a. Due to uncertainties, the peg may hit somewhere other than the desired goal, as in the two cases shown in Fig. 1b and c, respectively. If the peg in Fig. 1c only leans very slightly towards the wall, then the contact may not be distinguishable from the one in Fig. 1b due to sensing uncertainties (e.g., position/orientation sensing uncertainties). Nevertheless, the recovery strategies for the two cases should be different. The recovery motion for the case in Fig. 1b can simply be a compliant translation, while for the case in Fig. 1c, the recovery motion should also involve rotation. Thus, the two cases have to be distinguished. On the other hand, not all the details about a contact are important to recovery motions. For example, the cases shown in Fig. 1b, Fig. 1d, and Fig. 1e are different in terms of the relative locations of the objects in contact and the precise topological relationships among the surface elements of those objects. Nevertheless, the recovery motions of those contacts may follow the same strategy — a compliant translation along the surface of contact towards the hole.

The research directly targeted to contact detection in the presence of uncertainties can be found in the work by Desai etc.[3, 4] and by Spreng[6]. Both approaches are of hypotheses-and-tests kind, i.e., testing the validity of certain contact hypotheses to obtain the correct contact information. Desai's method, in particular, first assumed a set of possible contact formations (between two objects), and then used force/moment equilibrium conditions with the force/moment sensory data to eliminate certain contact formations. However, the key problem of how to obtain the contact hypotheses (i.e., initial contact formations) remains intact. Spreng's method used position/orientation sensing data to hypothesize about a contact in terms of motion freedoms and test motions for verification. The method, however, seems to be limited to 2D cases. Moreover, the use of test motions may cause new failures and further complicate the situation.

Although recovery motions are apt to be compliant to be less sensitive to uncertainties, the effect of uncertainties must still be taken into account in order to ensure successful implementations of the desired compliant motions. For example, in order to push the object in Fig. 2a along the surface successfully, the applied force must be in proper direction (w.r.t. the normal **N** of the surface) and magnitude to overcome the friction as well as to keep the object always in contact with the surface. This, however, has to be achieved in the presence of the orientation sensing uncertainty in **N**, the force/moment sensing uncertainty, and the modeling/control uncertainty in the force controller. So far the problem has not been addressed in the literature.

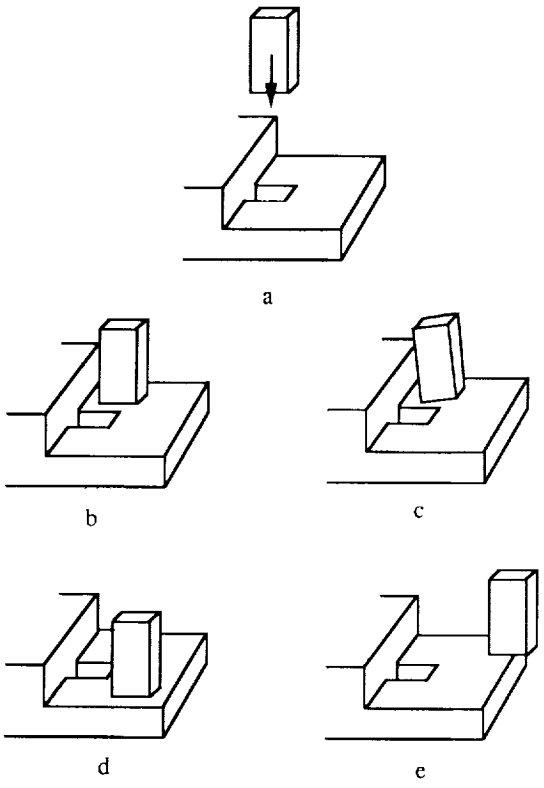


Figure 1: A Peg-in-Hole Example

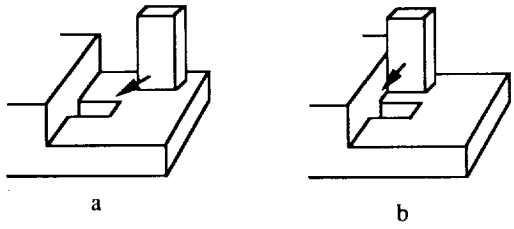


Figure 2: Compliant Translations

This paper will first discuss what kind of contact information is *enough* for planning recovery motions, an issue that has not been addressed in previous research, and define *recovery-oriented* concepts of contact. Then it will present a framework of using different sensors, especially position/orientation and vision sensors, to compensate each other in order to obtain the contact information of desired precision in spite of sensing uncertainties. It will also explain how to ensure the success of compliant motions by imposing proper force/moment constraints on the commanded force/moment applied to the held object (by the robot) and certain design constraints on the nominal and uncertainty parameters of the system.

2 Contacts and Assumptions

Since the contact detection problem mainly deals with unexpected *interactions* between the object held by a robot and the environment which, in most cases, is known approximately, we can assume that the environment is *fixed* in the sense that all parts or fixtures in the environment are pre-known; only the collisions between the held part and other parts can be unexpected. Thus, the problem can take advantage of a relatively stable and known environment in contrast to a robot navigation problem. We can also assume that the objects involved in an unexpected collision (i.e., the held object and some fixed objects in the environment) are in a static state, provided that there are force/moment guards to stop a robot motion once a collision occurs.

Now the concern is what kind of contact information will be needed in providing enough aid to the planning of recovery motions. From the example shown in Fig. 1, one can see that the detection of contact surfaces is surely important since they constitute the constraining surfaces for the compliant recovery motion. In addition, the basic topological formation of contact also matters since different formations may require different courses of recovery motion even if the contact surfaces are the same (as shown by the two contact cases in Fig. 1b and Fig. 1c). However, not all the details in the formation of a contact are important to recovery motions (e.g., the cases shown in Fig. 1b, Fig. 1d, and Fig. 1e share the same kind of recovery motion). Thus, we will introduce the concepts of contact which both facilitate detection and meet the need of recovery planning.

We will use the following topological exterior-elements of objects: faces, edges, and vertices in our descriptions. We define a *face* as a closed surface, i.e., a surface and its boundaries, and an *edge* as a closed edge line, i.e., an edge line and its boundary points[1]. Clearly, the exterior of a finite solid consists of finite faces with shared boundaries (among two or more faces). The boundaries of a face consist of edges, and the boundaries of an edge consist of (two) vertices. We say *two topological elements touch* iff the *interior* region of the two elements touch. Thus, we don't think that an edge touches a face if only a boundary point of the edge (i.e., a vertex) touches the face, and instead, we say that a vertex touches a face.

We now define a *principal contact* (PC) between two faces as one of the following: face-face (f-f), face-edge or edge-face (f-e or e-f), face-vertex or vertex-face (f-v or v-f), edge-cross (e-cross), edge-touch (e-touch), edge-vertex or vertex-edge (e-v or v-e), vertex-vertex (v-v) (Fig. 3), such that if there are more than one pair of topological elements (from the two faces respectively) that touch each other, the PC is determined by the pair in which the two topological elements are not the boundaries of the topological elements (of their respective faces) in the other pairs in touch. Now a *contact formation* (CF) can be introduced to define a contact between two objects, as a set of PC's involved (e.g., $\{< f_1^1, f_3^2 >, < e_4^1, f_1^2 >, ...\}$)[2].

[1] Formal definitions of *surfaces, edge lines*, and *vertex points* can be found in[10].

[2] This definition is quite different from that in[4, 3].

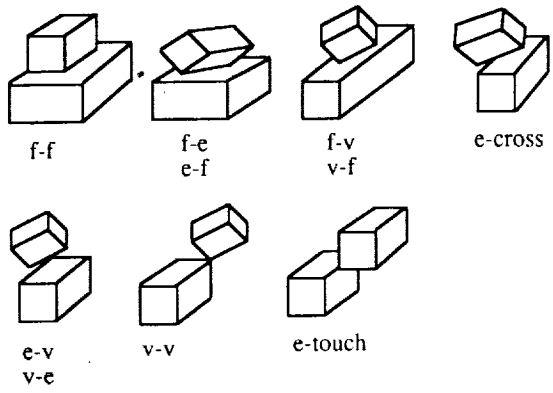


Figure 3: Principal Contacts between Two Faces

It is not difficult to observe that, for polyhedral objects, a PC of type f-f, f-e (or e-f), f-v (or v-f), or e-cross, involves only one *contact plane* (CP) — the tangent plane through the contact points determined by the principal contact. For non-polyhedral objects, a PC of type f-f, f-e (or e-f), f-v (or v-f), or e-cross, involves either a *contact surface* determined by the principal contact points or a *contact plane* tangent to the principal contact point(s).

It is, on the other hand, not generally proper to talk about a contact plane or surface for a PC of e-touch, e-v (or v-e), or v-v type, since there exists an infinite number of such contact planes or surfaces. However, PC's of types e-touch, e-v (or v-e), and v-v are rather purely mathematical concepts and rarely occur in reality due to their extremely unstable nature. Thus, we will not consider those types in this paper, assuming that they have zero probability of occurring. For simplicity, we will also restrict our discussion to polyhedral objects; thus *surfaces* are reduced to *planes* only, edge lines are reduced to *straight-lines*, and there is a contact *plane* associated with each PC.

3 Contact Detection

Let's consider an unexpected collision result in a contact between the object held by the robot obj_h and one fixed object. In principle, if the location of the held object can be sensed, then the contact formation can be deduced or derived from that sensed location, the boundary representations of both objects in the CAD model-base, and the pre-known location of the fixed object. However, in practice, the sensed location is often different from the actual location due to sensing uncertainty; thus, the contact formation derived can be wrong, or the derivation yields no contact at all. To solve the problem, our proposed strategy is to first obtain all *possible* contact formations based on the current locations sensed about the two objects, taking into account position/orientation sensing uncertainties, and then to use vision sensing to reduce the set of contact formations and to obtain satisfactory information about the contact. Force/moment or other sensing methods can also be included in the system.

3.1 Position/Orientation Sensing

First, the fixed object can be identified fairly accurately based on the sensed location of obj_h, since the contact was due to the motion deviation of obj_h from a *preplanned* path[3], and the deviation is generally small. Suppose the fixed object identified is obj_f. Then, the location of obj_f can be obtained from the pre-stored database. Let f_f, e_f, and v_f denote the face, edge, and vertex items of obj_f respectively, which are described with respect to the coordinate system of obj_f. Given the location of obj_f in the reference coordinate system of the workspace, i.e., the world coordinate system, those descriptions can then be easily transformed to be with respect to the world coordinate system. Similarly, let f_h, e_h, and v_h indicate the face, edge, and vertex items of obj_h, described in the coordinate system of obj_h. Based on the sensed location of obj_h in the world coordinate system, those descriptions can be transformed to be with respect to the world coordinate system.

Now we need to examine how the information of the given location of obj_f and the *sensed* location of obj_h can contribute to the detection of the contact formation between obj_f and obj_h. Let ϵ_p denote the *position sensing uncertainty*, such that for any point P, $\|\mathbf{P}^a - \mathbf{P}^s\| \leq \epsilon_p$, where $\mathbf{P}^a$ and $\mathbf{P}^s$ are the actual and the sensed positions of P. Let the angle ϵ_o denote the *orientation sensing uncertainty*, such that for any vector $\mathbf{N}$, $\angle(\mathbf{N}^a, \mathbf{N}^s) \leq \epsilon_o$, where $\mathbf{N}^a$ and $\mathbf{N}^s$ are the actual and the sensed vectors. Obviously, the uncertainties ϵ_p and ϵ_o in the location of obj_h affect the descriptions of f_h, e_h, and v_h items in the world coordinate system and thus the determination of the spatial relationships between those surface elements of obj_h and the surface elements of obj_f. Fig. 4 gives an example showing the ambiguity in determining a PC due to ϵ_p and ϵ_o. It is not difficult to observe that while there are many possible PC's, the possible contact planes involved are fewer. In other words, contact planes are relatively robust and insensitive to position/orientation sensing uncertainties. Therefore, we use position/orientation sensing to reason about contact planes first. The objective is to determine all *possible* contact planes, and for each contact plane, all possible PC's that may contribute to it, based on the given location of obj_f, the sensed location of obj_h, and the position/orientation sensing uncertainties ϵ_p and ϵ_o.

The detection of possible contact planes can be done by checking the relationship between a face of obj_h and a face of obj_f for all possible pairs of such faces between the two objects. Consider a face f_h^i of obj_h and a face f_f^j of obj_f,

[3]Therefore, the sequence of the objects adjacent to the path is known.

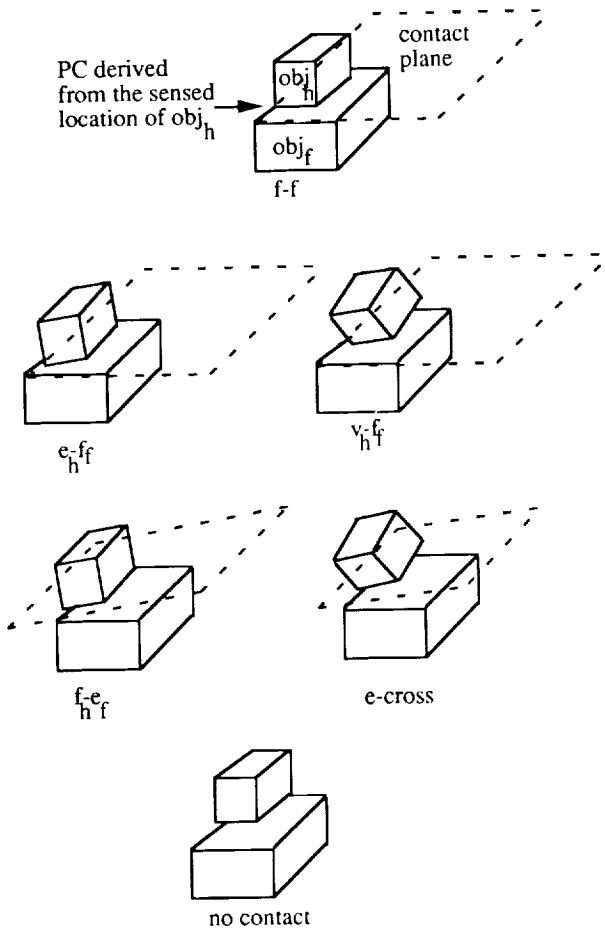


Figure 4: The Uncertainties in PC due to ϵ_p and ϵ_o

which lie on planes p_h^i and p_f^j respectively. If the projection of f_h^i on p_f^j possibly intersects f_f^j, taking into account ϵ_p and ϵ_o, then we can check if the minimum distance d_{min} between the sensed f_h^i and f_f^j is greater than ϵ_p or not. Fig. 5 shows examples of different spatial relationships between f_h^i and f_f^j and the corresponding d_{min} for each case. If $d_{min} > \epsilon_p$, we can conclude that there is no contact between f_h^i and f_f^j. Otherwise, there exists the possibility of a contact between f_h^i and f_f^j. The next step is to determine all the possible contact planes and PC's between the two faces. Our strategy is to construct models of all possible PC's by virtually (not physically) conducting the following operations on f_h^i and f_f^j:

- TOUCH — translate f_h^i (or f_f^j) along $\mathbf{d}_{min}$ a distance d_{min} to make f_h^i (or f_f^j) touch f_f^j (or f_h^i),
- TILT — tilt f_h^i (or f_f^j) about some axis on p_f^j coinciding an edge or vertex of f_h^i (or f_f^j).

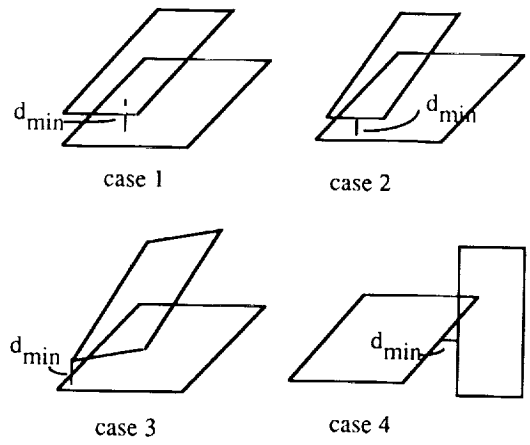


Figure 5: d_{min} between f_h^i and f_f^j

Specifically, if $d_{min} \leq \epsilon_p$, TOUCH will be conducted to make f_h^i contact f_f^j, and a PC can be determined by the orientations of f_h^i and f_f^j. The relationships between f_h^i and f_f^j can be distinguished in the following ways (Fig. 5):

1. $f_h^i \parallel f_f^j$;
2. case 1 does not hold, and some edge e of one face is on the plane of another face;
3. both 1 and 2 do not hold, and all vertices of one face are on the same side of the plane of the the other face;
4. none of 1, 2 and 3 hold.

Based on which case holds, the PC can be of types f-f, e-f or f-e, f-v or v-f, and e-cross respectively.

Based on the result of TOUCH, which gives a contact plane and a PC, TILT can be applied to vary the orientations of f_h^i and f_f^j within the orientation sensing uncertainty bound ϵ_o so that other possible PC's and contact planes can be obtained. Fig. 6 shows some examples. Note that the fundamental issue about TILT is *how to tilt* in order to get all possible PC's. There are generally an infinite number of ways of tilting f_h^i or f_f^j with the variations of orientation maintained within the range of ϵ_o. However, since the variations can only result in a finite number of PC's, just such number of tiltings will be sufficient. The definition of TILT above reflects this observation. For example, if the initial PC is $< f_h^i, f_f^j >$, we can tilt f_h^i or f_f^j along all its edges and each line through one of its vertices on the contact plane which is not collinear to the two edges forming the vertex; then we will obtain all possible e-f (or f-e) and v-f (or f-v) types of PCs. Based on each e-f (or f-e) PC, if the edge intersects an edge of the other face, then by tilting about the latter edge, a possible e-cross PC can be obtained.

By considering all possible face pairs between obj_h and obj_f in the way described, while trying to avoid or eliminate redundancy, the possible contact formations between obj_h and obj_f can be obtained. The final result would contain a set of possible contact planes (CP), and for each CP, a set of possible PC's that may result in the CP.

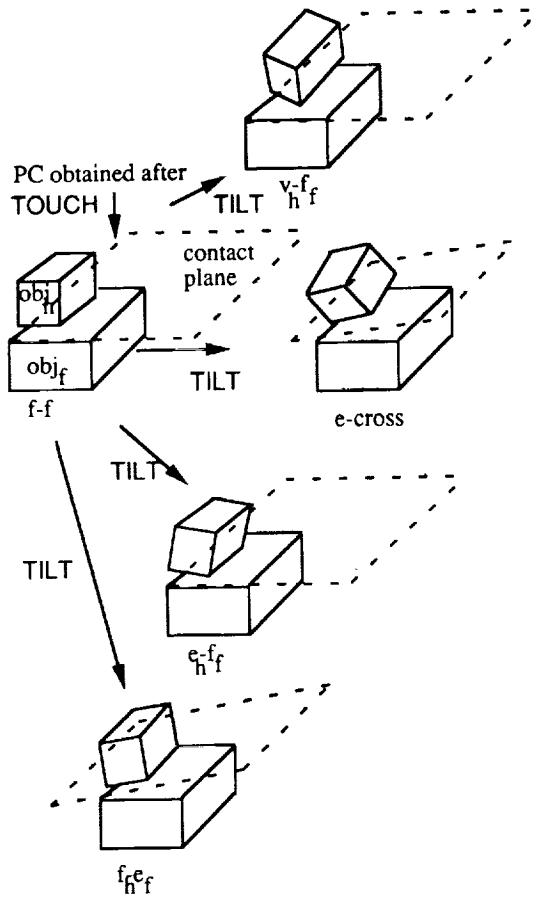


Figure 6: Possible PC's obtained by TILT

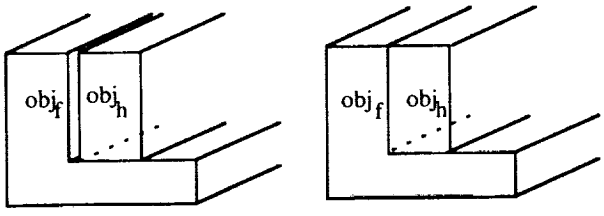


Figure 7: The Ambiguity in Contact Formation

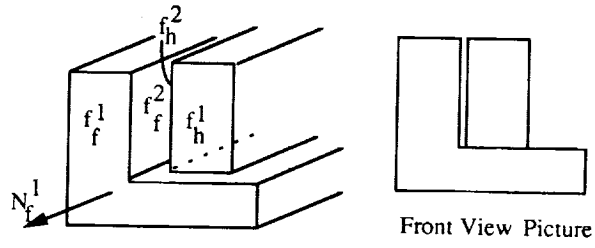


Figure 8: A Picture to Show if f_f^2 and f_h^2 Are in Contact

3.2 Vision Sensing

Using vision sensing to identify contacts is attractive in that the image information of a contact can convey most directly the *topological meaning* of the contact, i.e., the contact formation. As introduced previously, due to the position/orientation sensing uncertainties, it is difficult to determine whether a basic contact formation between obj_h and obj_f really exists given the location of obj_f and the sensed location of obj_h. For example, in Fig. 7, it is impossible to know which contact formation the contact is really in from position/orientation sensing only, if the horizontal distance between obj_f and obj_h is smaller than ϵ_p. By vision sensing, however, the problem could be solved *if certain picture(s) could be taken properly and reasoned effectively.* For example, if a picture can be taken from the direction opposite to the normal of f_f^1, then whether f_f^2 and f_h^2 are in contact can be determined by checking whether f_f^2 touches f_h^2 in the image (Fig. 8). Obviously, the following issues are important in using vision:

- how to view the contact, i.e., how to place the camera and take a picture;
- how to separate the features of interest from the rest of the things in an image;
- how many different views should be taken in order to obtain sufficient information about a contact.

The information obtained from position/orientation sensing is essential for dealing with the above issues. Recall that from position/orientation sensing, all possible contact planes can be obtained and for each contact plane, all possible PC's that may contribute to the contact plane can also be obtained. This information not only defines the goal of vision sensing: to eliminate non-existing contact planes and the non-existing PC's, but also provides clues on how to do it.

To view a contact, pictures can be taken based on each contact plane and the associated PC's which are the result of processing position/orientation sensing data. We can assume that a camera is held by another robot hand so that it can be placed in different locations easily. Then the topological surface elements that appear in a picture and contribute to a PC (and the contact plane) can be extracted

from the picture by combining image segmentation/labeling techniques *with the 3D modeling information and the sensed position/orientation* of those elements. Since the processed image will only contain simple surface elements in the forms of 2D edges and vertices, it will be easy to reason about their relationship. With several images taken from different views, one can expect to obtain sufficient information about a PC. In the following paragraphs, we will describe a strategy to detect a PC using vision. By this strategy, what we want to know from an image will be a simple fact such as whether the concerned surface elements of two objects are in a line contact, in a point contact, or in no contact, and that information can be quite reliable in spite of noise (or uncertainties) in the image. We assume that proper thresholds can be easily found based on the size of the objects and the (bounded) noise to determine whether a contact is a *line* or a *point*.

To check if a given contact plane really exists, one can place the camera in a way such that the image plane is perpendicular to the contact plane (see Fig. 8, where the contact plane is determined by f_f^2 and f_h^2). By processing the image so that it contains only the edges of f_f^2 and f_h^2, whether the contact plane really exists can be determined easily.

To eliminate the wrong types of PC's from a given set of possible PC's of a contact plane, one can take pictures for each possible PC and check if the result is as predicted. If not, the PC can be eliminated. Specifically, to confirm a f-f PC $< f_h^i, f_f^j >$, four pictures can be sufficient (Fig. 9):

- *pic*1 — taken along the contact plane in a direction orthogonal to an edge of f_h^i;
- *pic*2 — taken along the contact plane in the direction orthogonal to the direction of picture 1;
- *pic*3 — taken along the contact plane in a direction orthogonal to an edge of f_f^j;
- *pic*4 — taken along the contact plane in the direction orthogonal to the direction of picture 3.

If all the pictures (which should be segmented and labeled as described previously) show that the contact region between the face elements of f_h^i and f_f^j forms a *line*, then $< f_h^i, f_f^j >$ is confirmed. In some cases two pictures can be sufficient. For example, if there is no f-e (or e-f) type of PC's for the given contact plane, then we only need *pic*1 and *pic*2 to confirm the PC $< f_h^i, f_f^j >$. To confirm a f-e (or e-f) PC $< f_h^i, e >$ (or $< e, f_f^j >$), the following two pictures can be sufficient (Fig. 10):

- *pic*1 — taken along the edge e on the contact plane;
- *pic*2 — taken along the direction perpendicular to e along the contact plane.

If *pic*2 shows a *line* contact region between the relevant face elements of the two objects, while *pic*1 shows an approximate *point* contact (or a *much shorter* line contact as the effect of orientation sensing uncertainty in e), then $< f_h^i, e >$ (or $< e, f_f^j >$) is confirmed. Similarly, two pictures are sufficient to confirm a f-v (or v-f) PC $< f_h^i, v >$ (or $< v, f_f^j >$). If two pictures are taken along the contact plane in orthogonal directions towards the vertex v (Fig. 11), and the contact regions shown on both pictures are points, then the PC $< f_h^i, v >$ (or $< v, f_f^j >$) is confirmed. As for an e-cross PC, since it will not share a contact plane with other type of PC's, if the contact plane exists, the PC is confirmed.

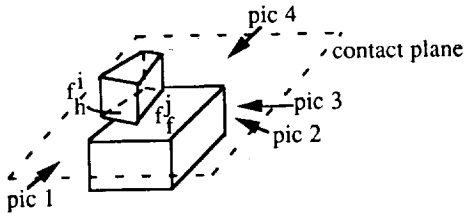


Figure 9: Four Pictures to Confirm a f-f Type PC

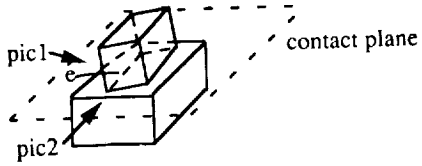


Figure 10: Two Pictures to Confirm a f-e or e-f Type PC

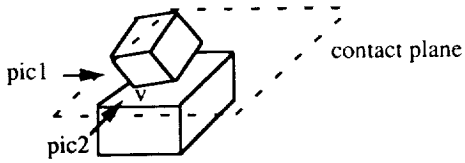


Figure 11: Two Pictures to Confirm a f-v or v-f Type PC

4 Integration of Other Sensors

We have shown that by using vision to eliminate the non-existing contact planes and the non-existing PC's, the exact contact formation can be determined from the set of possible contact formations initially obtained from position/orientation sensing. However, the major limitation of vision sensing lies in the possible occlusion of certain PC's, especially when the objects are non-convex. Thus, other sensing means, such as force/moment and tactile sensing[3, 4][1, 2] may also be needed, which can be readily added in this stage.

5 Compliant Motions for Error Recovery

A compliant recovery motion of the held object from an unexpected contact can be automatically planned[11] with the detection of the contact formation and the contact planes involved, as well as other information, such as the desired path of the held object and its current (sensed) location.

For polyhedral objects (as assumed in Section 2), there are the following basic types of compliant motions:

- translations constrained by one plane or two planes (Fig. 2),
- frictional rotations (Fig. 12a),
- non-frictional point-constrained and line-constrained rotations (Fig. 12b),
- combined frictional/non-frictional rotation (Fig. 13a), and
- combined translation/frictional-rotations (Fig. 13b and c).

It is necessary to determine then, in the presence of uncertainties (as introduced in Section 1), proper forces and moments to be applied to the held object by the robot, so that, based on the contact information (contact formation and contact planes), each type of the above motions can be implemented successfully in spite of uncertainties.

For *pure* compliant translations and rotations (as listed above), detailed analysis can be found in[9, 8] in which the proper forces and moments are determined in terms of force and moment constraints involving uncertainty bounds and under certain design constraints of system parameters. Note that the orientation sensing uncertainty is modeled as ϵ_o in Section 3.1. of this paper. The imperfections associated with force/moment sensing, modeling, and control are modeled simply as *force/moment control uncertainties*, ϵ_{ff} and ϵ_{mm}, which are defined as the maximum possible difference in magnitude between a desired or commanded force and the actual force applied and the maximum possible difference in magnitude between a desired or commanded moment and the actual moment applied respectively.

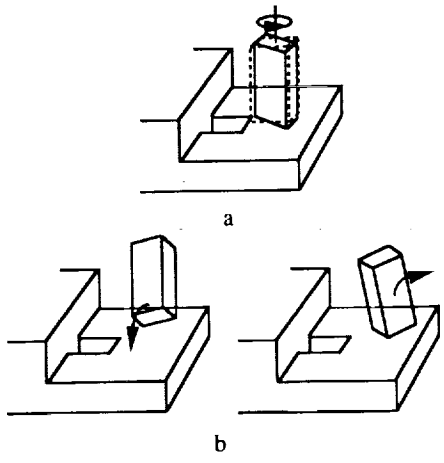


Figure 12: Compliant Rotations

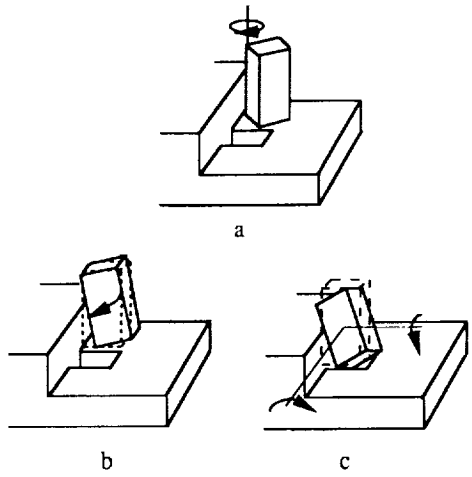


Figure 13: Combined Compliant Motions

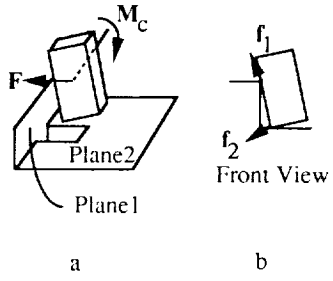


Figure 14: Force/Moment for a Combined Motion of Translation/Frictional-Rotation

In the same fashion, constraints can be derived for combined compliant motions, for example, a combined motion of translation and frictional rotation (as depicted in Fig. 13b). Such a motion can be implemented by applying a force $\mathbf{F}$ and a moment $\mathbf{M_c}$ on the center of the object as shown in Fig. 14a. The combination of $\mathbf{F}$ and $\mathbf{M_c}$ will result in compliant translations of the contact points of the held object against frictions[4], where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the equivalent forces generating the compliant translations. However, due to ϵ_o in the sensed normal $\mathbf{n}_1^s$ of plane 1 and ϵ_{ff}, the actual applied force $\mathbf{F}^a$ may not be parallel to the actual normal $\mathbf{n}_1^a$. Similarly, due to ϵ_o and ϵ_{mm}, the actual applied moment $\mathbf{M}_c^a$ may not be exactly parallel to the actual intersection line of plane 1 and plane 2. It is thus necessary to distinguish the force/moment components that will generate the desired translation/frictional-rotation from the components that may cause undesirable motions of the held object. Upon the force/moment components for generating the translation/frictional-rotation, constraints can be derived involving ϵ_o, ϵ_{ff}, ϵ_{mm}, and the friction coefficient μ, which when satisfied, guarantee that no sticking will occur and that the motion will always be compliant (i.e., contacts will always be maintained). On the other hand, upon the force/moment components that may cause undesired motions, constraints (also involving ϵ_o, ϵ_{ff}, ϵ_{mm}, and μ) can be derived so that when they are satisfied, the effect of friction will prevent the undesirable motions from occurring. By synthesizing the two sets of constraints obtained, proper constraints on the magnitudes of the commanded $\mathbf{F}$ and $\mathbf{M}$ can be obtained, as well as possible design constraints on ϵ_o, ϵ_{ff}, ϵ_{mm}, μ, and other object-related parameters (such as those characterizing the shape and size of the held object). Upon the satisfaction of the design constraints, and by chosing proper $\mathbf{F}$ and $\mathbf{M}$ based on the force/moment constraints, the desired translation/frictional-rotation can be achieved in spite of uncertainties. As for the force/moment control to implement a desired force/moment (which is determined by our force/moment constraints), many approaches can be found in the literature, as have been surveyed and compared by Hollerbach[5] and Whitney[7].

6 Conclusions

This paper studied the effect of uncertainties in recognizing failures of robot motion in the forms of contacts and in implementing contact-based recovery motions. It proposed recovery-oriented contact detection using multiple sensing modalities and outlined means to ensure successful contact motions in spite of uncertainties. The research, however, is in its early stage. Further development and testing of the idea are necessary in the future. Subjects of special importance includes studying how accurate vision information will be and uncertainties in vision, investigating further integrations of different sensors in the system, and testing the existing and new results.

[4] Let μ be the friction coefficient of the materials, and assume the Coulomb friction cone model.

References

[1] P. K. Allen. *Robotic Object Recognition Using Vision and Touch.* Kluwer Academic Publishers, 1987.

[2] Paolo Dario. Contact sensing for robot active touch. In M. Brady, editor, *Robotics Science*, pages 138–163. MIT Press, 1989.

[3] R. Desai. *On Fine Motion in Mechanical Assembly in Presence of Uncertainty.* PhD thesis, Department of Mechanical Engineering, the University of Michigan, 1989.

[4] R. Desai, J. Xiao, and R. Volz. Contact formations and design constraints: A new basis for the automatic generation of robot programs. In B. Ravani, editor, *NATO ARW: CAD Based Programming for Sensor Based Robots.* Springer-Verlag, July 1988.

[5] J. M. Hollerbach. Kinematics and dynamics for control. In M. Brady, editor, *Robotics Science*, pages 378–431. MIT Press, 1989.

[6] M. Spreng. Dealing with unexpected situations during the execution of robot motions. In *Proc. IEEE International Conference on Robotics and Automation*, 1991.

[7] D. E. Whitney. Historical perspective and state of the art in robot force control. In *IEEE Int. Conf. Robotics and Automation*, 1985.

[8] J. Xiao. *On Uncertainty Handling in Robot Part-Mating Planning.* PhD thesis, Department of EECS, the University of Michigan, 1990.

[9] J. Xiao. Force/moment constraints for robot compliant motions in the presence of uncertainties. In *IEEE Int. Symposium on Intelligent Control*, August 1991.

[10] J. Xiao. On recognition of contacts between objects in the presence of uncertainties. In *SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing*, April 1991.

[11] J. Xiao and R. Volz. On replanning for assembly tasks using robots in the presence of uncertainties. In *IEEE Int. Conf. Robotics and Automation*, 1989.

N93-11957

EXOS RESEARCH ON MASTER CONTROLLERS FOR ROBOTIC DEVICES

Beth A. Marcus, Ph.D, President
Ben An, Mechanical Engineer

EXOS, Inc., 8 Blancard Road
Burlington, MA 01803

Brian Eberman

Massachuestts Institute of Technology
Artificial Intelligence Lab
77 Massachusetts Avenue
Cambridge, MA 02139

1 ABSTRACT

Two projects are currently being conducted by EXOS under the Small Business Innovation Research (SBIR) program with NASA. One project will develop a force feedback device for controlling robot hands, the other will develop an elbow and shoulder exoskeleton which can be integrated with other EXOS devices to provide whole robot arm and hand control. This paper will cover the project objectives, important research issues which have arisen during the developments, and interim results of the projects.

The Phase I projects currently underway will result in hardware prototypes and identification of research issues required for complete system development/integration.

2 INTRODUCTION

There has been a significant amount of research on robot hands (Salisbury, 1985) and force reflecting controllers (e.g. Bejczy and Salisbury, 1983, Agronin, 1987) for robot arm manipulation. The Stanford/JPL 6 DOF hand controller is an example of a compact non-antropomorphic force reflecting arm master. There have been a variety of anthropomorphic or semi anthropomorphic force reflecting masters for hands or arms (review in Iwata 1990) but little work has been done to integrate manipulation and sensing for teleoperation of dexterous robot hands and arms together. One complete system (Jacobsen 1989) employs an anthropomorphic configuration which is bulky, is not compatible with the field environment, and is very expensive. In addition there are many research issues which arise in the design and development of these devices which have not been addressed in the literature to date.

Very few force reflecting sytems have been developed for generating forces on the individual fingers and the palm. Iwata (Iwata 1990) describes a system for the thumb, two fingers and the palm. The palm is actuated by a six degree-of-freedom parallel stage driven by electric motors. Each of the two fingers and thumb has single degree-of-freedom motion and is also driven by electric motors. This system can transmit large forces to the hand and fingers, but is limited to pinch type grasps because of the single degree-of-freedom plates for the fingers.

Burdea, Zhuang, and Roskos (1991) describe a system based on pneumatic cylinders and the VPL dataglove. This sytem uses pnumatic cylinders placed between the inside of the palm and the fingertips to generate forces. The system is compact and light, but at present can only simulate grasp forces between the palm and the fingertips. Contact of the fingers with objects supported externally cannot be simulated.

The current EXOS projects are the first steps towards building a comfortable, lightweight, field compatible and affordable integrated anthropomorphic force feedback master for teleoperation of dexterous robot hands and arms. This process of development, by necessity, includes answering some basic research questions regarding human capabilities, limitations, and perceptions. This paper will report on progress to date towards these objectives.

3 SENSING AND FORCE REFLECTING EXOSKELETON (SAFiRE) PROGRESS

This program is intended to result in the design and development of a sensing and force reflecting exoskeleton (SAFiRE) which provides control signals to robot hands and force feedback to the human operator. The SAFiRE will allow robot hands working in unstructured environments to gently touch objects, and finely manipulate them without exerting excessive forces.

3.1 Phase I Project Objectives

The goal of the Phase I effort is to build a 2 degree of freedom (DOF) prototype SAFiRE which demonstrates the feasibility and utility of a SAFiRE in controlling robots.

3.2 Initial Design Goals

We surveyed the relevant literature, discussed the overall objectives an developed a set of screening criteria for the designs we were developing Each of the members of the design team then ranked the criteria. The results were that comfort, ease of use and feel were considered to be the most important criteria. The criteria and their rankings are given below.

3.3 Specifications

We then spent some time attempting to develop and quantify specifications for the details of the device. Some of the key areas have little available information on the performance required in order to achieve the design goals. In these cases the consensus was to examine systems which have been built, how they performed, and how similar the application they were used in is to our application. From this type of analysis we can make a best guess specification which can be revised as the program progresses and hardware is available for conducting experiments.

The following is a preliminary specification for the SAFiRE. The 2 DOF prototype will meet a subset of these specifications.

1. Hand Size: 50th %ile female to 95th %ile male

2. Joints:
   DOF: 10 with force reflection, others with

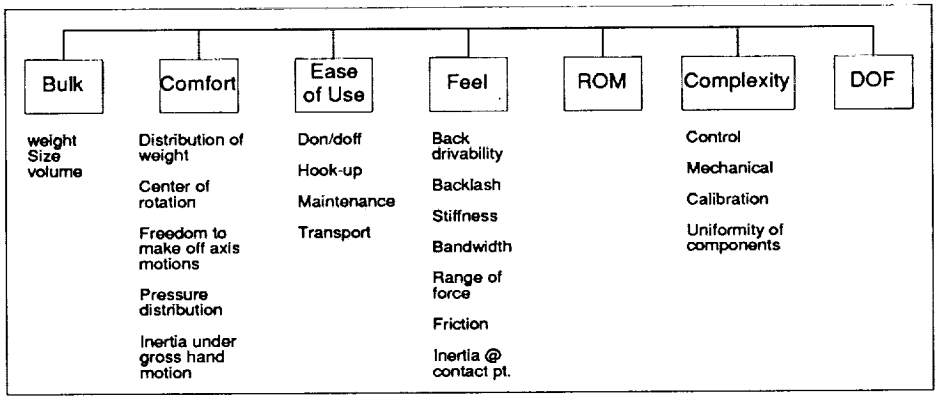


Figure 1: SCREENING CRITERIA

RANKING

CRITERA	RANK	SCORE	PERCENT
FEEL	1	17.66	25.0
COMFORT	2	16.15	22.9
EASE OF USE	3	11.94	16.9
BULK	4	9.92	14.1
COMPLEXITY	5	8.81	12.5
ROM	6	3.00	4.3
DOF	7	3.00	4.3

position only Range of Motion: 120o maximum for an individual joint Angular Resolution: ~1/2o

3. Force Reflection:
   Direction: Both Flexion and Extension
   Magnitude: 4 lbf-in Peak torque at MP joint 2 lbf-in Continuous

4. Dynamic Performance:
   Friction: Less than .25 oz-in
   Backlash: Less than .20°
   Frequency Response: 3Db point at ~50 Hz

5. Weight:
   Exoskeleton: 1/2 oz. Per DOF
   Actuators: mounted on forearm ~1 oz. Per actuator

These specifications will be used as a guideline to compare concepts. As we develop prototypes and conduct experiments we will be able to refine these specifications.

3.4 Design Options

In developing a design which fits the screening criteria a variety of concepts for transmission and actuation were considered. The following charts describe these options:

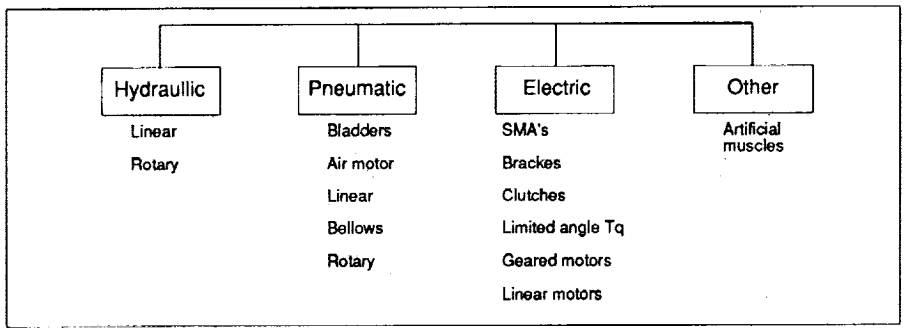


Figure 2: ACTUATION OPTIONS

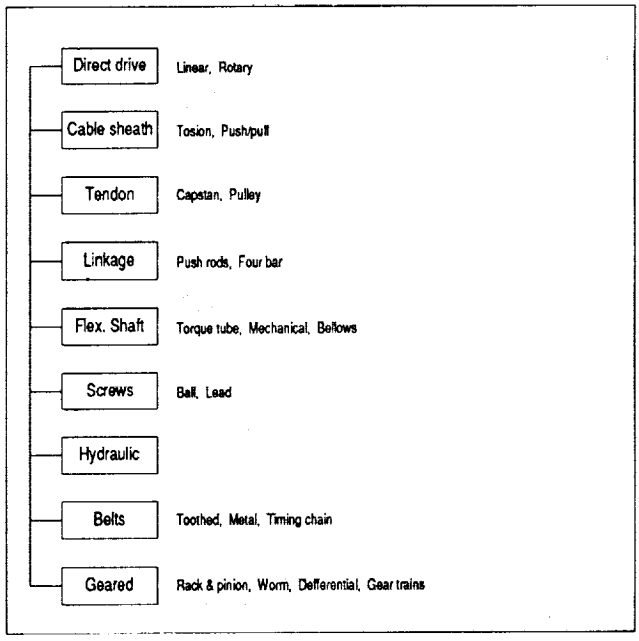


Figure 3: TRANSMISSION OPTIONS

Several overall concepts for the system configuration were developed and used to evaluate the actuation and transmission concepts. Mock-ups and bench top experiments were performed to assist in the process of determining which options were feasible. A 2DOF working prototype of the resulting design is currently being detailed and constructed.

3.5 Research Issues

A number of issues were identified which were not covered adequately enough in the literature to permit the results to be used in developing screening criteria, specifications and finally designs. Specific areas of research needs include:

Types of Feedback
- -tactile
- -kinesthetic (force)
- -auditory

Feedback
- -intensity
- -location
- -transitions

Grounding
- -to the hand
- -to ground
- -to other body parts

The types of specific research projects that will help to resolve these issues include:

When designing any force feedback device the objectives in terms of the desired bandwidth, force output, gain, and resolution must be determined. To determine the bandwidth required one must look at the task and determine what objects need to be manipulated, their stiffness and other mechanical properties, but one must also determine the range of stiffness of the finger (for feedback to the hand) and other mechanical properties of the hand and arm. The properties of the hand and arm vary from person to person, but they also vary based upon the state of the different muscles being used.

Imagine how it feels when your arm is locked with all its muscles contracted when your arm strikes an object as compared with the feeling associated with a completely relaxed arm encountering an object. Consider the example of arm wrestling. If your muscles are contracted when the competition begins it requires a much larger force to move your arm than if they are relaxed. Imagine the differences in the design of a arm wrestling telerobot if the competitor always had relaxed muscles versus one which had to deal with contracted muscles, or what about the more realistic case where the exact conditions of the competitors arm are unknown. To size the motors, set their required response time and many other factors the range of human responses to force feedback must be known.

To develop the transmission, actuation, and control principles the amount of perceptible friction and backlash must be quantified. The human brain is able to learn, and adapt very readily to variations between the real object and the simulated one in certain cases without perceiving the difference. But which types of departures are acceptable and how much deviation from the exact physical conditions to be simulated is possible without reduced performance needs to be studied. For example, virtually all mechanical mechanisms which are practical for feedback to the hand have some friction. If the feedback device is being operated in a free motion mode (i.e. the simulated hand is not touching any objects) the operator will perceive that friction as a feeling of touching an object. The large the friction the harder the object will be perceived to be. For different parts of the hand and body as a whole, the level of perceptible friction (i.e. friction induced force) is unknown. In addition the inertia of the device will feel like an object.

Very little work has been done on the ratio of the force at the master to that of the slave. This ratio is very task dependant. For example a robot (or virtual robot) which can lift a 1000 pound payload clearly does not want to feedback all that force to the operator. On the other extreme, in a microsurgical application the level of force required to make a tiny incision may be beneath the level of perception of the human hand, therefore in this situation the force must be amplified. Some situations require both extremes and thus the issue of transitions must be considered. In addition when scaling the force different phenomena may need to be scaled differently. For example, if you do microsurgery the major things that you may want to feel could be surface tension and drag which are not felt at all a human scale. Therefore a strategy for the representation must be studied and developed for each application.

No known studies have been found which address the issues associated with perception of force as it relates to where the device is grounded. Most developers of feedback hardware have just decided to ground it to external ground or to the hand. We conducted some crude experiments in conjunction with this

program which told us that grounding the device to the arm has the potential to provide realistic feedback within certain constraints. It is necessary to understand how to use grounding to optimize task performance.

Auditory feedback has been used to assist in providing a more realistic work environment. Only anecdotal reports of accidental improvements in performance due to auditory feedback of, for example, motor torques has been reported. A systematic study of auditory feedback to determine when it is appropriate, how to use it effectively, and how much of a performance improvement it provides in different task situations should be conducted.

Little work has been done to evaluate where tactile feedback could be best used, the type of information to be presented and how it works in combination with force and auditory feedback.

In the initial phase of this project we have conducted some preliminary experiments on these issues and will be better able to study them once the Phase I prototype is completed.

4 EXOSKELETAL ARM MASTER (EAM) PROGRESS

The exoskeleton arm master (EAM) program will result in an elbow and shoulder measurement system which will provide precise measurement of the arm and forearm motion. The system will be portable and comfortable to wear, thereby allowing free movement of the operator without loss of control or precision in robot arm position.

4.1 Phase I Project Objectives

The goal of the Phase I effort is to build an EAM which demonstrates the feasibility and utility of an EAM in providing integrated control capabilities for whole-arm manipulation and grasping.

4.2 Arm Kinematics

The human body is a complex linkage with many small motions in addition to the major joint motions which make the task of attaching an exoskeleton to the arm comfortably while obtaining accurate, reliable and repeatable measurements of joint motions difficult. The seven basic human arm motions which must be converted into robot control signals are illustrated in Figure 4.

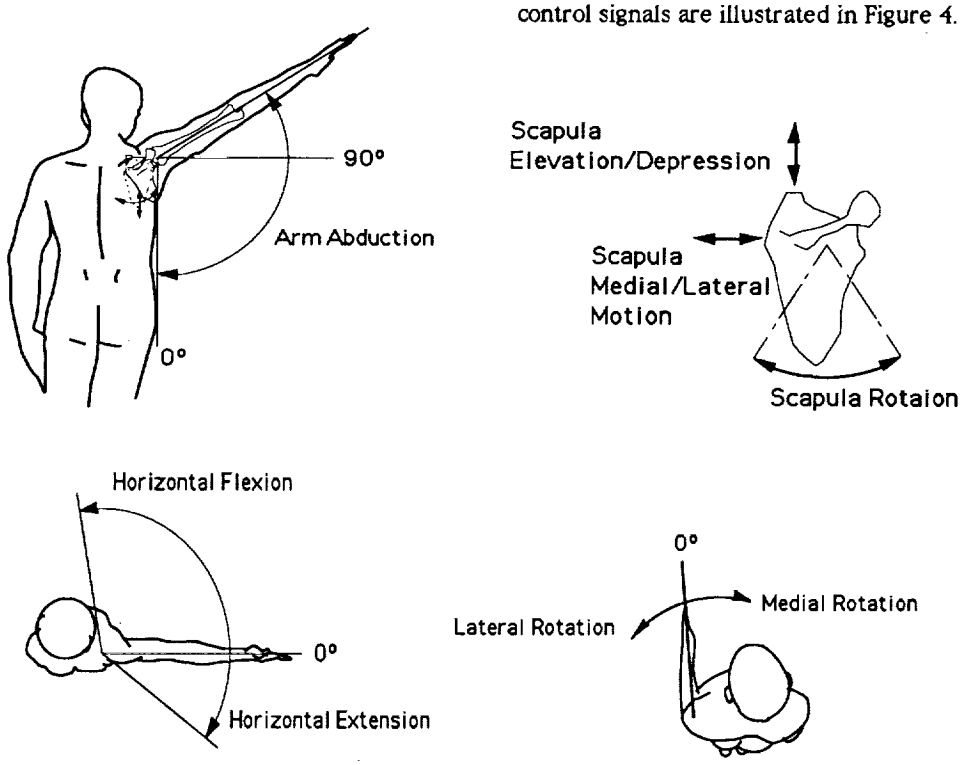


Figure 4. Major shoulder movement. (after Kapandji I.A.)

4.2.1 Shoulder

The shoulder is a complex structure which involves a combination of motions from several joints (Figure 4.). For a sample, arm abduction as shown in Figure 4., uses the glenohumeral joint to provide motions during the initial portion of the range of motion. Once the arm moves past the horizontal position, shoulder shrugging (Scapula elavation and rotation) is employed. In addition the scapula has medial and lateral movement. Robot arms do not typically have these motions. Therefore in developing an exoskeleton master for controlling robot arms the motion must either be transduced, filtered out, or used with a method of transforming the motions into robot arm motions.

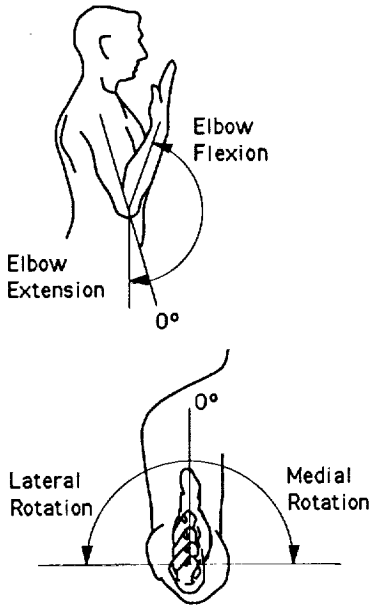


Figure 5. Major elbow motion. (after Kapandji I.A.)

4.2.2 Elbow

The elbow and forearm have two degrees of freedom which must be measured. Elbow flexion is a straight forward motion with only a small amount of motion of the joint center of rotation to be accommodated. The challenge in this joint is how to comfortably and accurately attach to the body to follow this motion. It is even more difficult to address attachment with respect to the supination/ pronation because this motion is accomplished by the radius pivoting around the ulna

4.2.3 Wrist

The wrist has two degrees of freedom which must be measured. In addition the wrist has several small motions which make it difficult to transduce the major wrist motions. The center of rotation moves, the two motions are coupled in some ranges and the configuration of the bones to which one must attach changes with motion. The EXOS GripMaster™ transduces wrist flexion/extension and radial/ulnar deviation. It uses the patented Dexterous HandMaster™ passive pivots and general linkage configuration to allow accurate tracking of these two motions.

Although the initial EAM prototypes will concentrate on the elbow and shoulder, the GM wrist and DHM hand can be used as is for completing the control.

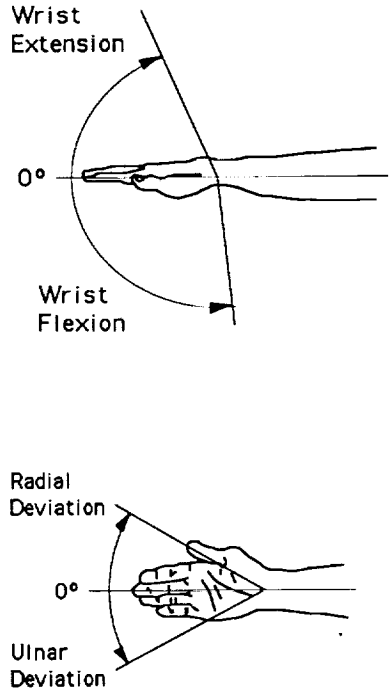


Figure 6. Major movement of the wrist. (after Kapandji I.A.)

4.3 Required Ranges of Motion

Although the exact ranges of motion of the shoulder and elbow joints is some what controversial [An, Doody, Freeman, Inman, Lucas, Morrey, NASA-STD-3000, Steindler and Youm]. The following table summarizes the commonly cited data from medical literature and those published by NASA. Those starred must be transduced for robot control.

The fewer sensors a given design has the lighter it will be and lower the inertia. However, since it is not possible to make the axes of motion of the human joint align exactly with the axes of motion of the exoskeleton for all users with a small number of sensors the computation required to transform the sensor readings into human joint angles increases. Therefore trade-offs were conducted among these factors to arrive at an initial design configuration.

JOINT MOTION	NORMAL ROM†	SUITED ROM††
Shoulder Abduction/Adduction*	150°	150°
Shoulder Media/Lateral Rotation*	130°	120°
Shoulder Horizontal Flexion/Extension*	170°	150°
Scapula Elevation/Depression	10-12cm	N/A
Scapula Medial/Lateral Movement	15cm	N/A
Scapula Rotation	60°	N/A
Elbow Flexion/Extension*	145°	130°
Forearm Supination/Pronation*	180°	180°
Wrist Flexion/Extension*	170°	N/A
Wrist Radial/Ulnar Deviation*	60°	N/A

† Kapandji I.A. (1982)
†† NASA-STD-3000. (1989)

4.4 Concept Development

Once the human motions are known the task of determining the optimal linkage to follow these motions must be determined. There are several competing design considerations that were identified in the process of developing concepts. They are:

- minimization of number of sensors
- minimization of computation com plexity
- Low inertia
- rigidity of attachment
- comfort
- ease of donning/doffing
- adjustability to accommodate human variability
- range of motion
- safety

4.5 Testing

In order to test the ability of the EAM to measure the human jointsaccurately, an additional accepted method of measuring the human joints is required. There is no "gold standard" for human motion measurement. Data derived form video or other optical techniques rely on placing markers on the surface on the body. This method has inherent errors due to skin motion and shape changes in the limbs due to muscular contraction. X-ray data can tell you where the bones are, but without placing radiopaque markers in the bone correlation of one view with another has significant errors. Therefore we decided to begin with a known motion which was produced by building a shoulder simulator. This device is simply three high precision potentiometers with bearings (0.1%

linearity) to ensure the highest repeatability possible with off the shelf components. These were mounted such that their axes of motion coincide and correspond to the three primary shoulder degrees of freedom. Provision for rigidly attaching the EAM to the shoulder simulator was also made to eliminate another factor affecting the measurements in the real world, motion artifact due to motion of the attachments. A schematic of the device is given below.

Thus various kinematic configurations and algorithms for transforming the measured angles to human joint angles can be tested against a "gold standard." A similar device is being constructed to test wrist designs.

5 CONCLUSIONS

Currently the feasibility demonstration prototypes for the SAFiRE and EAM are being built and tested. An array of research issues and design trade offs have been identified and initial estimates to resolve these issues or conduct trade-offs have been made to develop the first level prototypes. Once these devices are complete testing will be required to determine how well these devices meet the design goals and to identify areas requiring more detailed research and development to produce fully functional devices.

References:

1. Mason, M. T. and Salisbury, J. K., **Robot Hands and the Mechanics of Manipulation**, MIT Press, Cambridge, Massachusetts, 1985.
2. Bejczy, A. K., & Salisbury, J. K., Jr., "Controlling Remote Manipulators Through Kinesthetic Coupling", **Computers in Mechanical Engineering**, (July 1983), 48-60.
3. Agronin, M. L., " The design of a Nine-string Six-degree-of-freedom Force-feedback Joystick for Telemanipulation", **Proceedings of the NASA Workshop on Space Telerobotics**, Pasadena, CA 1987, pp. 341-348.
4. Iwata, H., "Artificial Reality with Force-feedback: Development of Desktop Virtual Space with Compact Master Manipulator", **Computer Graphics**, 24:4, August 1990.
5. Jacobs, S.C, 'The State of the Art in Dexterous Robotics," Conference Dinner Speaker, IEEE International Conference on Robotics and Sutomation, May 1989
6. Burdea, G., Zhuand, J., Roskos, E., **Towards a Portable Dexterous Master with Force Feedback, Presence: Teleoperators and Virtual Environments**, MIT Press, January 15, 1991.
7. An, K.-N., Browne, S.K., Tanaka, S., Morrey, B.F., "Three-Dimensional Kinematics of Glenohumeral Elevation.", **J. Orthop. Res.**, Vol. 9, No. 1, 1991.
8. Doody, S.G. and Waterland, J.C., "Shoulder Movement During Abduction in the Scapular Plane", **Arch. Phys. Med. Rehabil.** 51:595, Oct., 1970
9. Freeman, L. and Munroe, R.R., "Abduction of the arm in the scapular plane: Scapular and glenohumeral movements.", **J. Bone Joint Surg.**, 48A(8):150, dec., 1966.
10. Inman V.T., Saunders, J.B., and Abott, L.C., "Observations of Function of the Shoulder Joint", **J. Bone Joint Surg.** (Br) 26:1, 1944
11. Kapandji I.A., **The Physiology of the Joints**. Translated by Honore L.H. Churchill Livingstone, Leith Walk, Edinburgh, 1982.
12. Lucas D.B., "Biomechanics of the Shoulder Joint", **Arch. Surg.**, 107:425, Sept., 1973.
13. Morrey, B.F. and Chao, Y.S., "Passive Motion of the Elbow Joint", **J. Bone Joint Surg.** 58(A):4, 1976
14. National Aeronautics and Space Administration, **Man-Systems Integration Standards**, NASA-STD-3000, Vol. I, Rev. A, 1989.
15. Norkin, C.C., Levangie, P.K., **Joint Structure and Fuction: A Comprehensive Analysis**, F.A. Davis Company, Philadelphia.
16. Steindler, A., **Kinesiology of the Human Body**, Charles C Thomas, Springfield, Ill., 1955.
17. Youm, Y., et. al., "Biomechanical Analysis of Forearm pronation-supination and Elbow Flexion-Extension", **J. Biomech.**, 12:245, 1979.

This work was carried out on the sponsorship of:

1. NASA Marshall Space Flight Center Contract: NAS8-38910
2. NASA Johnson Space Center Contract: NAS9-18452

N93-11958

Generation of 3-D Surface Maps in Waste Storage Silos Using a Structured Light Source*

B. L. Burks, J. C. Rowe and M. A. Dinkins
Oak Ridge National Laboratory†

B. Christensen and C. Selleck
Sandia National Laboratory

D. Jacoboski and R. Markus
Westinghouse Materials Company
of Ohio

Surface contours inside the large waste storage tanks typical of the Department of Energy (DOE) complex are, in general, highly irregular. In addition to pipes and other pieces of equipment in the tanks, the surfaces may have features such as mounds, fissures, crystalline structures, and mixed solid and liquid forms. Prior to remediation activities, it will be necessary to characterize the waste to determine the most effective remediation approaches. Surface contour data will be required both prior to and during remediation. In this presentation, the authors will describe the use of a structured light source to generate 3-D surface contour maps of the interior of waste storage silos at the Feed Materials Production Center at Fernald, Ohio. The landscape inside these large waste storage tanks bears a strong resemblance to some of the landscapes that might be encountered during lunar or planetary exploration. Hence, these terrestrial 3-D mapping techniques may be directly applicable to extraterrestrial exploration. In further development, it will be demonstrated that these 3-D data can be used for robotic task planning just as 3-D surface contour data of a satellite could be used to plan maintenance tasks for a space-based servicing robot.

*Research sponsored by the U. S. Department of Energy, ER&WM Office of Technology Development.

†Managed by Martin Marietta Energy Systems, Inc., for the U. S. Department of Energy under contract DE-AC05-84OR21400.

Session A3: ERROR DETECTION AND CONTROL

Session Chair: Joe Herndon

FUZZY LOGIC CONTROL OF TELEROBOT MANIPULATORS

Ernest A. Franke and Ashok Nedungadi
Robotics and Automation Department
Southwest Research Institute
6220 Culebra Road
San Antonio, TX 78228-0510

ABSTRACT

Telerobot systems for advanced applications will require manipulators with redundant degrees-of-freedom that are capable of adapting manipulator configurations to avoid obstacles while achieving the user specified goal. Conventional methods for control of manipulators (based on solution of the inverse kinematics) cannot be easily extended to these situations. Fuzzy logic control offers a possible solution to these needs.

A current research program at Southwest Research Institute has developed a fuzzy logic controller for a redundant, 4 degree-of-freedom (DOF), planar manipulator. The manipulator end point trajectory can be specified by either a computer program (robot mode) or by manual input (teleoperator mode). The approach used expresses end-point error and the location of manipulator joints as fuzzy variables. Joint motions are determined by a fuzzy rule set without requiring solution of the inverse kinematics. Additional rules for sensor data, obstacle avoidance and preferred manipulator configuration, eg. "righty" or "lefty", are easily accommodated. The procedure used to generate the fuzzy rules can be extended to higher DOF systems.

INTRODUCTION

Telerobots, mechanical manipulators that are controlled by an operator from a remote location and are also capable of automatically performing programmed tasks, will become increasingly important in the future as more complex and demanding applications are attempted. Telerobot applications typically occur in locations where direct access by humans is restricted by the remoteness (undersea and space) or by the environment (nuclear or chemical waste sites). In contrast to industrial robotic applications, the workcell for these telerobot tasks is unstructured and the exact operations required are not known in advance. Operator guidance and control of the remote manipulator is necessary but at the same time there will be some operations that occur so often it is desirable to provide the capability for automated execution of specific task elements upon operator command. In many instances the operator will maneuver the telerobot manipulator into the desired position and then initiate a programmed task element such as scanning, cutting, turning or grasping.

Current and anticipated applications for telerobots will be difficult to accomplish with typical approaches to manipulator kinematics and control. One capability that will be important in future telerobotic designs is kinematic redundancy, or the use of additional degrees-of-freedom in the mechanical structure of manipulator. The great majority of robot and telerobot systems in use today are non-redundant so that they are constrained to reaching a specified end position in only one geometric orientation. Kinematic redundancy provides several advantages including:

- Obstacle avoidance - the manipulator can reach around objects in the workspace and still achieve the desired endpoint position.
- Fail functional - the manipulator can continue to operate in spite of the failure of a motor or gearbox although the advantages of kinematic redundancy may be lost.
- Configuration for tasks - different tasks such as pushing, pulling, turning, or grasping can be performed more efficiently with different manipulator configurations. Kinematic redundancy provides a means of selecting a desired configuration for a specific task.

In spite of the advantages, there are several reasons why kinematic redundancy is not in greater use. The use of additional joints increases both the cost and the complexity of manipulators. In addition, control of redundant systems requires selecting one configuration or path from a great number of possibilities instead of simply following the single path that is possible in the case of non-redundant systems. Approaches to the control of these systems have been based on optimization techniques and are not suited for real time use due to excessive computational requirements. [1]

In order to meet the requirements of future applications, an advanced telerobotic system should be capable of [2]:

- End point control - The manipulator should be able to position itself as required to reach an end point specified by the operator without requiring the operator to specify the alignment of each manipulator link.
- Obstacle avoidance - The manipulator should be able to sense obstacles in the workspace, store their locations, and select manipulator configurations to avoid obstacles without direct operator instructions.
- Sensor based control - Sensors should be included to monitor variations in the task

(process) being performed, variations between workpieces, variations in the workcell, and variations in the manipulator itself. Control of the telerobot should include provisions to automatically compensate for variations when possible or for allowing corrections to be made by the operator when necessary.

- The controller should have the capability for selecting manipulator configurations to enhance particular operations. This should include the ability to utilize joint angle combinations that provide the maximum force or torque to be exerted, configurations that minimize manipulator inertia during moves, and configurations that will be most convenient in making the transition to the next operation.

CONVENTIONAL MANIPULATOR CONTROL

Typical teleoperator manipulator control provides the operator with a direct control signal to each axis of the remote manipulator. In a master-slave arrangement, control signals from each axis of the local master unit are used to drive the corresponding axis of the remote (slave) manipulator. This requires that the two units have the same number of axis and correspond kinematically. The most common arrangements approximate the geometry of the human arm. The use of direct correspondence between master and slave axis control signals makes it difficult to accommodate redundant kinematics and does not allow the controller to implement sensor based operations or adaptive configuration control.

An alternative approach is to provide direct joint control of the remote manipulator by joysticks or other signal input devices. This eliminates the need for kinematic correspondence (since there is no master unit) but coordinated motions are more difficult to achieve. Joystick control of a remote unit requires an interface to inform the operator of the location and configuration of the manipulator, something that is best accomplished with graphic presentations. In addition, joystick control methods retain the one-to-one control signal to axis drive arrangement that does not allow implementation of adaptive motion or configuration control.

Conventional robot control systems are based on a mathematical model of manipulator kinematics. Measurement of axis variables allows calculation of the manipulator end-point location by solving the forward kinematic model of the manipulator. Motion control is accomplished by solving the inverse kinematic equations to determine the axis values needed to position the end-point at specified locations. This can be computationally difficult since the equations of motion are generally nonlinear and the inverse solution may not be unique for all points along a desired path. In order to cope with these difficulties, designers have resorted to linearizing the kinematic models, building relatively massive, slow robots, and avoiding redundant kinematic configurations. Conventional approaches to robot control make it very difficult to provide adaptive motion control such as modifying the path dynamically to adjust offset and orientation to a workpiece as might be needed to perform a surface inspection using an eddy current probe. Conventional approaches also are unable to cope with redundant kinematic geometries since the mathematical model of the system (including solution of the inverse kinematic equations) is too complex to permit real-time solutions.

MODEL-FREE APPROACHES TO ROBOT CONTROL

Even cursory consideration of biological motion control systems shows that the approach is completely different from that of conventional robot control. When we reach for an object, we determine the approximate error (distance from our hand to the object) and move in such a way as to reduce the error. We do not precompute the path or the elbow or shoulder angles that will be required to grasp the object. Our motions are directed toward the goal of continually reducing the distance between our hand and the object. This goal directed strategy makes it very natural to incorporate adaptive elements such as path changes to conform to a surface or to track a moving object. In fact, we are very successful at reaching and grasping both stationary and moving objects and apparently accomplish these feats without an accurate mathematical model of the kinematics involved.

Recent developments in neural networks and fuzzy logic have shown that non-biological systems can also perform feedback control without the use of an accurate model of the process or motion involved. This is accomplished by the use of neural networks or fuzzy associative memories that "estimate continuous functions from data without specifying mathematically how outputs depend on inputs" [3]. Control by neural networks and by fuzzy logic is similar in many ways. From the implementation standpoint, neural systems encode information in an unstructured form and typically are taught the function to be estimated by examination of a series of examples. In all but the simplest cases, when a neural network controller has been taught a particular function, the estimating function is distributed through the connections, summing nodes, and weights of the system. This makes it difficult to determine the effect that changes in the network will have on performance and also difficult to determine what changes should be made in the network to effect desired changes in performance. In many cases it is preferable to completely retrain the network to obtain modified performance rather than make incremental changes to the network structure or weights.

Fuzzy logic control systems encode knowledge in a much more structured way. Rather than being taught from

examples, fuzzy logic systems are typically implemented by drawing on knowledge of system operation to construct the cause-and-effect relationship rules entries. Specific input-output relations are expressed as Fuzzy Associative Memory (FAM) elements and the relationship between specific performance measures and FAM rules can be determined more easily than for neural network systems. This provides the designer with better understanding of the effect of changes in the rules and allows modification or extension of the rules to incorporate new performance requirements.

Although both neural network and fuzzy logic approaches can be used for manipulator control, the fuzzy logic approach was selected for the research reported here. The fuzzy logic approach allowed an initial control system to be derived from fundamental concepts without the need for extended training sets. It provided a better understanding of effect of changes in the controller structure and allowed beginning with examples of reduced dimensionality and generalizing the results to higher dimensions and kinematic redundancy. Finally, the fuzzy logic control approach includes the capability for adding additional rules to incorporate additional features such as sensor adaptive control, obstacle avoidance, and configuration modifications without requiring a new training set for these additions. [4]

The proposed fuzzy logic robot controller mimics intelligent human-like decision-making through a fuzzy rule base, which is essentially a collection of varying degrees of cause-and-effect relationships, such as: "If A is observed then perform control action B" or "If less of A is observed then perform less of control action B" or conversely "If more of A is observed then perform more of control action B". Since the fuzzy logic robot controller presented in this paper is based on linguistic rules, it does not require the derivation of the complex inverse kinematic equations.

The approach used to implement fuzzy logic control of a manipulator was to calculate the error between the actual manipulator end-point (given by solution of the forward kinematic equations) and the desired end-point specified by a trajectory generator. A set of FAM-rule matrices was derived for each manipulator axis to associate the controller inputs (the end-point error and joint positions) to the desired output (drive signals to the joint motor). The investigation began with simulations of controller performance for a two-axis planar robot and was then extended to three- and four-axis kinematically redundant planar robot simulations.

DEVELOPMENT OF THE RULE BASE

The procedure used for development of the fuzzy rule base is derived from the forward kinematic equation for a four DOF planar manipulator, shown in Figure 1.

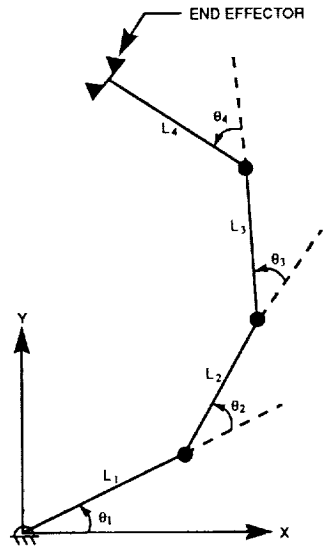


Figure 1. A Four DOF Planar Manipulator

Referring to Figure 1, the end point coordinates of the manipulator X_0 and Y_0 can be expressed as a function of the joint angles and link lengths L_1, L_2, L_3, and L_4.

$$\begin{aligned} X_0 &= L_1\cos(\theta_1)+L_2\cos(\theta_1+\theta_2)+L_3\cos(\theta_1+\theta_2+\theta_3)+ \\ &\quad L_4\cos(\theta_1+\theta_2+\theta_3+\theta_4) \\ Y_0 &= L_1\sin(\theta_1)+L_2\sin(\theta_1+\theta_2)+L_3\sin(\theta_1+\theta_2+\theta_3)+ \\ &\quad L_4\sin(\theta_1+\theta_2+\theta_3+\theta_4) \end{aligned} \quad (1)$$

Equation (1) describes the forward kinematic model for the four DOF planar robot of Figure 1. Taking the first variations of equation (1) from some nominal robot configuration θ_1, θ_2, θ_3, and θ_4 we have:

$$\begin{aligned} \delta x &= [\frac{\partial x}{\partial \theta_1}]\delta\theta_1+[\frac{\partial x}{\partial \theta_2}]\delta\theta_2+[\frac{\partial x}{\partial \theta_3}]\delta\theta_3+[\frac{\partial x}{\partial \theta_4}]\delta\theta_4 \\ \delta y &= [\frac{\partial y}{\partial \theta_1}]\delta\theta_1+[\frac{\partial y}{\partial \theta_2}]\delta\theta_2+[\frac{\partial y}{\partial \theta_3}]\delta\theta_3+[\frac{\partial y}{\partial \theta_4}]\delta\theta_4 \\ \delta x &= A_1\delta\theta_1+A_2\delta\theta_2+A_3\delta\theta_3+A_4\delta\theta_4 \\ \delta y &= B_1\delta\theta_1+B_2\delta\theta_2+B_3\delta\theta_3+B_4\delta\theta_4 \end{aligned} \quad (2)$$

where:

$A_1 = -[L_1 \sin(\theta_1) + L_2 \sin(\theta_1+\theta_2) + L_3 \sin(\theta_1+\theta_2+\theta_3) + L_4 \sin(\theta_1+\theta_2+\theta_3+\theta_4)]$

$A_2 = -[L_2 \sin(\theta_1+\theta_2) + L_3 \sin(\theta_1+\theta_2+\theta_3) + L_4 \sin(\theta_1+\theta_2+\theta_3+\theta_4)]$

$A_3 = -[L_3 \sin(\theta_1+\theta_2+\theta_3) + L_4 \sin(\theta_1+\theta_2+\theta_3+\theta_4)]$

$A_4 = -[L_4 \sin(\theta_1+\theta_2+\theta_3+\theta_4)]$

$B_1 = [L_1 \cos(\theta_1) + L_2 \cos(\theta_1+\theta_2) + L_3 \cos(\theta_1+\theta_2+\theta_3) + L_4 \cos(\theta_1+\theta_2+\theta_3+\theta_4)]$

$B_2 = [L_2 \cos(\theta_1+\theta_2) + L_3 \cos(\theta_1+\theta_2+\theta_3) + L_4 \cos(\theta_1+\theta_2+\theta_3+\theta_4)]$

$B_3 = [L_3 \cos(\theta_1+\theta_2+\theta_3) + L_4 \cos(\theta_1+\theta_2+\theta_3+\theta_4)]$

$B_4 = [L_4 \cos(\theta_1+\theta_2+\theta_3+\theta_4)]$

Since equation (2) represents the linearized kinematic model of the 4 DOF planar robot, the Principle of Superposition is valid when applied to the individual joint angle variations $\delta\theta_1$, $\delta\theta_2$, $\delta\theta_3$, and $\delta\theta_4$. Thus the combined contribution of $\delta\theta_1$, $\delta\theta_2$, $\delta\theta_3$, and $\delta\theta_4$ for a given δ_x and δ_y is equal to the individual contributions of $\delta\theta_1$, $\delta\theta_2$, $\delta\theta_3$, and $\delta\theta_4$. We assume that each of the four joint variations ($\delta\theta_1$, $\delta\theta_2$, $\delta\theta_3$, and $\delta\theta_4$), the desired move in the x and y directions and variables A_1, .. A_4 and B_1, .. B_4 of equation (2) are characterized by the following primary fuzzy sets: Large Positive (LP), Medium Positive (MP), Small Positive (SP), Small Negative (SN), Medium Negative (MN), and large Negative (LN). Inspecting equation (2) and applying the Superposition Principle, two Banks of Fuzzy Associative Memory rules (FAM) are proposed that together determine $\delta\theta_1$ for a given δ_x and δ_y. The rules relating required changes in θ_1 to desired end-point motion in the x-direction are shown in Figure 2. The rules for desired y-motion are the same except for use of the B_i coefficients rather than the A_i.

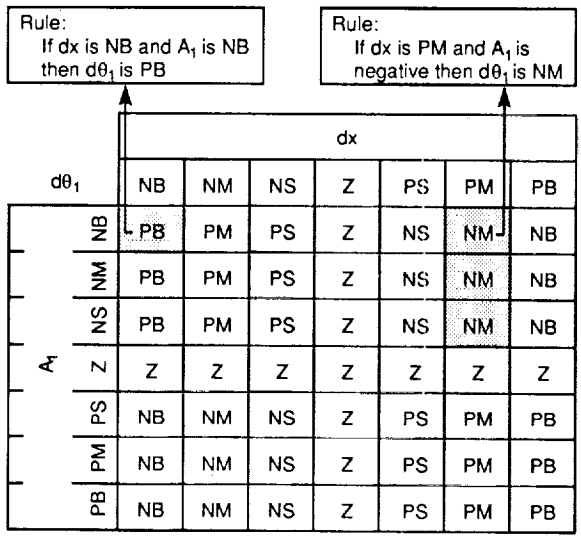


Figure 2. FAM Bank for Joint 1 (δ_x term).

The fuzzy rules for the remaining joints (θ_2, θ_3, θ_4) are similar. The entries of each FAM bank were filled through graphical inspection. For example, if the commanded move in the x direction were MP and A_1 were to MN, then a MN $\delta\theta_1$ is required to achieve the requested move. In a similar fashion, each of the entries of the above FAM Bank (A) can be filled out. Each entry in the FAM bank represents a fuzzy associative memory rule or input-output transformation of the form:

IF (A_1 is MN) AND (δ_x is MP) THEN $\delta\theta_1$ is MN

Thus FAM bank A is comprised of 7 x 7 = 49 rules. Inspecting the symmetry of the FAM bank, the following compound rules can be formulated:

IF (A_1 is LN) OR (A_2 is MN) OR (A_3 is SN) AND (δ_x is SN)
THEN $\delta\theta_1$ is SP

This construct reduces the 49 rules per FAM bank to 14 rules per table. Furthermore the overlapping (25%) of the seven primary fuzzy sets that describe A_1, δ_x, and $\delta\theta_1$ are such that a state (A_1, δ_x) can belong simultaneously to a maximum of 2 fuzzy sets, therefore only a maximum of 4 rules per FAM bank will have a non-zero contribution.

CURRENT RESEARCH

The fuzzy logic control scheme described above has been simulated using a PC-AT computer and a Fuzzy-C Development System from Togai Infralogic. Kinematic models of several planar manipulators were used to investigate performance for two, three, and four DOF systems. The fuzzy rule sets were translated into C code and program modules were added to generate end-point path trajectories and provide graphic display of the robot motion. Straight line and circular path trajectories were generated and the motion of the simulated manipulators was analyzed.

The simulation showed that the manipulators with the fuzzy controller were able to follow the specified trajectories. For two and three DOF manipulators, the results were similar to conventional control systems. The more interesting results were obtained for the four DOF, redundant robot case. For very small path steps, the actual path followed differed from the specified trajectory by only small amounts. As the step size was increased, the following error increased. For large step sizes (and correspondingly large errors) the system became unstable and the simulated robot left the commanded trajectory and wandered about the work space. Since analysis shows that the computational requirements for fuzzy control are much less than for conventional approaches of control of redundant manipulators, it should be possible to maintain a high servo update rate so that the step size will be small even for large velocities.

Bench mark tests comparing the execution speed of the fuzzy logic controller with the classical controller revealed that the approach described in this work was 1.5 times faster than traditional methods which requires solution of the inverse kinematic equations. This increase in performance was achieved mainly because the FLC did not solve any inverse kinematic equations and all internal evaluations of the fuzzy rule base was performed by integer additions and multiplications. On the other hand, traditional methods require several matrix manipulations such as inversions and multiplications. The tradeoff for the increased speed of the developed FLC is greater trajectory tracking errors compared to classical controllers that explicitly solve the inverse kinematic equations. This is typical of a fuzzy

rule base approach because a fuzzy rule base describes a patch in the state space rather than an exact single point. In several robotic applications, the reduced tracking ability may not necessarily be a drawback. There are numerous robot applications which do not require precise trajectory tracking, such as paint stripping, paint application, obstacle avoidance or applications that require the robot to move a payload from one point to another.

A noteworthy aspect of the FLC controller is that the total number of rules required to control the robot arm is linearly dependent to the total number of degrees-of-freedom. This implies that the scheme is still implementable for robots with higher degrees of freedom.

On inspection of the proposed fuzzy rules, it is evident that the contribution of the individual joints is evaluated independent of the other axis. The individual contributions are then combined using the Principle of Superposition to result in some motion at the end point of the robot. This implies that the individual axis motion of the robot can be decoupled and therefore evaluated independently and simultaneously resulting in a very straightforward parallel implementation with a single dedicated fuzzy chip for each axis of motion.

The fuzzy controller has also been used to control a small four DOF planar manipulator at SwRI. This robot had been built for research in control of redundant systems and includes a flexible VME-based general purpose minicomputer motion controller [5]. Fuzzy control was implemented by transferring the C code from the simulator to the VME computer system and developing program modules for the interface to the servomotor controllers. Preliminary tests of straight line trajectories demonstrate that the fuzzy controller is operating satisfactorily on this system.

Plans for future work include additional tests and performance measurements using the planar research robot. Preliminary investigations have indicated that rules for obstacle avoidance can be expressed as fuzzy associative memories and combined with the motion control FAM's. This concept will be simulated and then tested on the planar research robot. Other plans include extension of the FLC approach to spatial robots and development of a hardware implementation of the fuzzy controller.

APPLICATIONS

The results of this research project have far-reaching implications for control of both robotic and telerobotic systems. The ability to perform end-point control of a manipulator without the need for solving the inverse kinematic equations can lead to considerable reductions in computational requirements for robot and telerobot controllers. In addition, it has been noted that there is no interaction between fuzzy rules governing the motion of the individual manipulator joints. This allows the evaluation of the control signals for each joint to be performed independently and simultaneously and leads to a very straightforward parallel implementation of microprocessors. Figure 3 illustrates one possible block diagram for a controller based on fuzzy logic. In the telerobotic mode, operator inputs (from joystick or hand controller) will be used to specify the desired end point motion. The trajectory generator performs smoothing and interpolation to calculate the specified position at discrete time intervals. Simultaneously, a second processor receives the measured joint angles and calculates the position of each joint using the forward kinematic solution. In the simplest configuration, the actual position and the difference between the actual and desired positions provide inputs to the FAM Axis Controller processors. These processors, one for each axis of the telerobot, compute the control signals for all axis drives simultaneously.

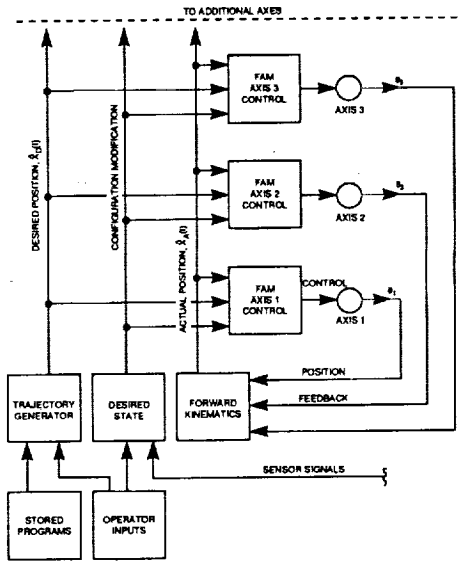


Figure 3. Block Diagram for a Telerobot Fuzzy Logic Controller.

The fuzzy logic controller can perform end point control of a redundant telerobot without requiring a kinematic correspondence between the master controller and the remote manipulator. This capability can be used in applications where redundancy is needed for fail-functional capability (operations on planetary space missions) or to provide maximum reach from a small ingress envelope (operations inside a nuclear waste storage vessel). In these cases the operator can use a simple hand controller to achieve the desired vector motions of the end point without having to be concerned about coordinated control of multiple manipulator joints.

An additional advantage of the fuzzy logic approach and the controller described above is that it provides a structure for adding additional fuzzy rules to provide

sensor-adaptive control, obstacle avoidance and other enhancements. As shown in the block diagram this is accomplished by adding signals that describe the desired state of the system and using these to drive additional fuzzy rules in each axis controller. This is most advantageous when applied to redundant manipulators since the configuration and positions of joints can be changed without affecting the position of the end point. In some telerobot applications the operator's control console would include switches to specify preferred configurations:

- elbow to right
- elbow to left
- use "zig-zag" configuration (for maximum force or torque)
- "open bow" configuration (for obstacle avoidance)
- configuration that keeps links near the manipulator base (to minimize inertia during moves)

Signals from each of these switches would activate or disable sets of fuzzy rules at each axis controller. The results of these fuzzy rule evaluations would be combined with the results of the position error rules in order to generate a command signal for each joint that would move the end point nearer the desired position (primary goal) while attempting to maintain the specified manipulator configuration (secondary goal). These techniques would be most effective for transport and assembly operations such as space station or subsea activities.

Signals from sensory devices can be used as inputs to fuzzy rule sets in the same way as signals from the operator's console. This provides a convenient extension to obstacle avoidance and adaptive path motion control. Obstacles in the workspace could be detected by sensors mounted on the manipulator links or by sensors having a global view of the workspace. Each axis controller would include fuzzy rules to move the manipulator links away from nearby obstacles. The obstacle avoidance signals would be combined with motion control and configuration preference signals to define the complete control signal. Similar techniques could be used to sense and maintain standoff distance from a surface or position relative to a seam. A telerobot equipped with these sensors would be useful for surface inspection since the operator could concentrate on maneuvering the end effector over the required surface confident that the fuzzy controller would maintain the correct standoff distance and prevent collisions with obstacles. Other applications include task level programming and navigation for autonomous mobile robots.

CONCLUSIONS

A non-algorithmic, model free approach has been developed that relies on a fuzzy rule base to evaluate the required axis motion to result in user desired end-point motion of the robot. This scheme does not require solution of the complex non-linear inverse kinematic equations to arrive at joint set points. This is in sharp contrast with traditional robot controllers. The fuzzy rule based controller 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 fifteen rules are required to evaluate individual joint axis motion and that a linear relationship exists between the number of rules and the degrees-of-freedom of the robot. This allows extension to higher DOF systems, including robots and telerobots with serial redundancy. Laboratory tests have demonstrated that FLC can be applied successfully to systems with kinematic redundancy and leads to implementation with far less computational requirements than conventional approaches. Finally, the FLC approach is very suitable for parallel processor implementation utilizing a dedicated fuzzy processor chip for each axis.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the Internal Research and Development program of Southwest Research Institute.

REFERENCES

[1] "New Inverse Kinematic Algorithms for Redundant Robots," T. C. Hsia and Z. Y. Guo, Journal of Robotic Systems 8(1), pp. 117-132, 1991 .

[2] "Sensor Integrated Control for Manipulators", E. Franke, V. Sturdivant, A. Nedungadi, proceedings of 1990 IEEE International Conference on Systems, Man, and Cybernetics, November, 1990.

[3] "Neural Network and Fuzzy Systems", B. Kosko, Prentiss Hall, Englewood Cliffs, June 1991.

[4] "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part II, C. C. Lee, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, March/April, 1990.

[5] "Investigation of Kinematically Redundant Serial Linkages", A. Nedungadi, Final Report for SwRI Internal Research Project 05-9508, 1989.

A New Scheme of Force Reflecting Control

Won S. Kim

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

ABSTRACT

A new scheme of force reflecting control has been developed that incorporates position-error-based force reflection and robot compliance control. The operator is provided with a kinesthetic force feedback which is proportional to the position error between the operator-commanded and the actual position of the robot arm. Robot compliance control, which increases the effective compliance of the robot, is implemented by low pass filtering the outputs of the force/torque sensor mounted on the base of the robot hand and using these signals to alter the operator's position command. This position-error-based force reflection scheme combined with shared compliance control has been implemented successfully to the Advanced Teleoperation system consisting of dissimilar master-slave arms. Stability measurements have demonstrated unprecedentedly high force reflection gains of up to 2 or 3 even though the slave arm is much stiffer than the operator's hand holding the force reflecting hand controller. Peg-in-hole experiments were performed with eight different operating modes to evaluate the new force-reflecting control scheme. Best task performance resulted with this new control scheme.

Introduction

In a typical telemanipulation system that does not support force reflection or compliance control, a stiff remote manipulator moves strictly according to a human operator's position command, and small errors between the actual and the commanded position of the manipulator can give rise to undesired large contact forces and torques. It is thus hard to expect safe and reliable telemanipulation with this system. Two major techniques that alleviate this excessive contact force problem are force reflection [2] and shared compliance control [9]. In force reflecting teleoperation, the operator can feel contact forces and torques through a force reflecting hand controller, and thus adjust the hand controller position naturally to reduce undesired contact force components. Experimental studies indicate a significant enhancement in the human operator's task performance with force reflection [5]. In shared compliant control, the operator's commanded position is altered by a compliant control force feedback in the robot side. This local autonomous force feedback in the robot side adds active compliance and damping to the stiff robot hand, making the robot more compliant to the environment and softening mechanical contacts/collisions between the manipulator and objects. Recent experiments demonstrated that shared compliant control is essential in time-delayed telemanipulation [9].

Recently orbital replacement unit (ORU) changeout experiments were performed with the JPL/NASA telerobot testbed system [7], and the experimental results showed that without shared compliant control (SCC) or force reflection (FR), the operator could not complete the task, while with SCC or FR the operator could perform the task successfully with reduced contact forces both in magnitude and duration. The results also indicated that the task performance with SCC was superior to that with FR in terms of task completion time, cumulative contact force, and total contact duration. The relatively poor performance with FR was mainly due to a poor force reflection gain. The maximum force reflection gain attainable without causing instability was only approximately 1/10. With this low gain, the operator could feel only 1 *lb* when the manipulator hand senses a 10 *lb* contact force. It is shown in this paper that the problem of poor force reflection is not specific to this system, but rather inherent to the conventional force reflection control scheme being used for dissimilar master-slave systems where the slave system usually has much higher stiffness than the effective stiffness of the human hand holding the force reflecting hand controller.

A major advantage of FR is that the operator actually feels the contact forces/torques sensed by the telerobot hand. This paper addresses two important issues related to FR: i) a new scheme of force reflecting control that makes high force reflection possible, and ii) assessment of the performance enhancement by providing the operator with both FR and SCC. Recently we developed a new scheme of force reflecting control that enables a sufficiently high force

reflection gain (up to 2 or 3) by utilizing position error and active compliance. This new scheme of FR combined with SCC is described in this paper. We also performed peg-in-hole experiments with eight different operating modes to evaluate this newly developed control scheme, and the results are presented.

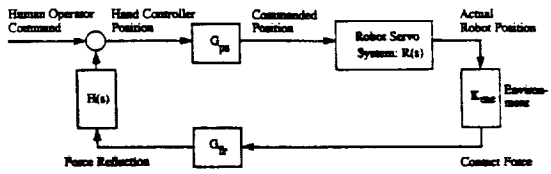


Fig. 1. A typical force-reflecting scheme for dissimilar master-slave arms.

Implementation of Position-Error-Based Force Reflection

In a typical force-reflecting telemanipulation system consisting of dissimilar master-slave arms, the position of a slave arm (remote manipulator) is controlled by the human operator through a master arm (force-reflecting hand controller) (Fig. 1), while the contact forces/torques sensed by the force/torque sensor at the base of the robot hand are reflected back to a human operator through the master arm. This forms a closed-loop system, and raises a stability issue. Our experience with the existing force-reflecting systems supporting dissimilar master-slave arms [7],[9] has shown that the force reflection gain from the robot hand to the force reflecting hand controller is limited to approximately 1/10. Namely, the operator can feel only 1 *lb* when the robot hand senses 10 *lb*. We now investigate this poor force reflection problem.

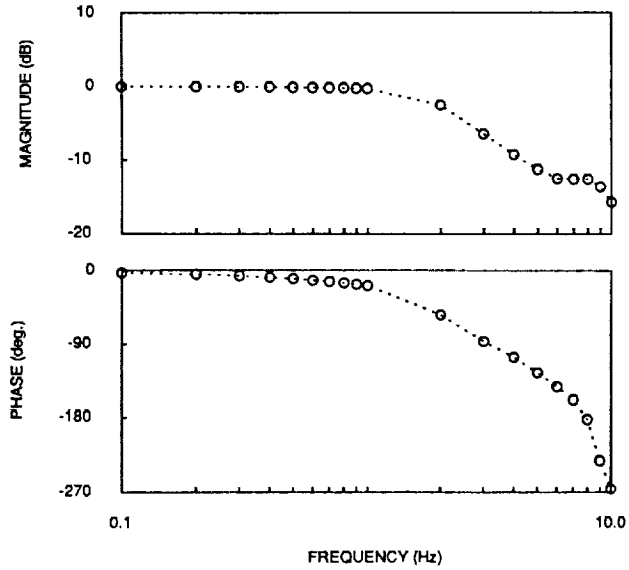


Fig. 2. A typical cartesian space frequency response of the PUMA arm used in our Advanced Teleoperation System

As a first-cut rough approximation, we assume a linear decoupled system model in cartesian axis. In Fig. 1, the open-loop transfer function Q(s) is given by

$$Q(s) = G_{ps} G_{fr} K_{me} H(s) R(s), \qquad (1)$$

where G_{ps} is the position command scale factor, G_{fr} is the force reflection gain, and K_{me} is the effective stiffness which is a parallel combination of the manipulator stiffness and the environment stiffness. $R(s)$ is the robot servo system transfer function in cartesian space [6],[8] and is given by a linear sum of the six second-order joint servo transfer functions with the DC gain of $R(0)$=1. $R(s)$ could be second-order, forth-order, or higher depending upon the cartesian axis and the arm configuration. An example of a cartesian space frequency response [8] of the PUMA arm used in our Advanced Teleoperation system [2] is shown in Fig. 2. In this example, the double-pole corner frequencies are at about 3 and 6 *Hz*, behaving as a fourth order system. $H(s)$ is the transfer function of the operator's hand holding the 6-degree-of-freedom force-reflecting hand controller [1]. The transfer function can be obtained by measuring the magnitude ratio of the hand controller deflection to the applied force input for different frequencies. Measurements indicate that the compliance value C_h (=$H(0)$) varies from about 1.0–2.0 *in/lb* (0.5–1.0 *lb/in* stiffness) with a loose grasp to about 0.1–0.2 *in/lb* (5–10 *lb/in* stiffness) for a firm grasp. The bandwidth of $H(s)$ is about 1 *Hz* for a loose grasp, and 3 *Hz* for a firm grasp. Typical frequency responses of the operator's hand holding the force reflecting hand controller for firm grasp (circle) and for loose grasp (triangle) are shown in Fig. 3 In order to have a stable teleoperation system with a constant force reflection gain G_{fr}, the open-loop DC gain $Q(0)$ should not be much greater than 1, since a higher loop gain causes instability due to the higher order

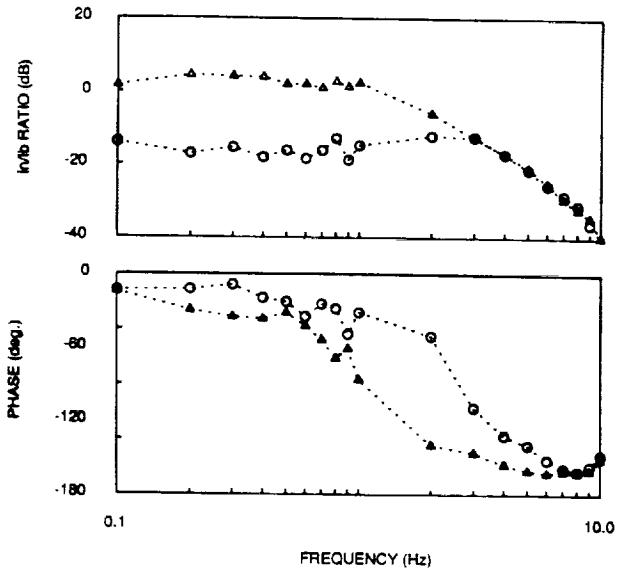


Fig. 3. Typical frequency responses of an operator's hand holding a 6-axis force-reflecting hand controller for firm grasp (circle) and for loose grasp (triangle). The magnitude ratio of the hand controller deflection to the applied force is plotted as a function of frequency.

dynamics of $H(s)R(s)$. Namely,

$$G_{fr} \le \frac{1}{G_{ps} K_{me} C_h}. \qquad (2)$$

In our typical system, the combined stiffness of the manipulator and environment is measured $K_{me} = 25\ lb/in$, and we assume that the operator's hand can maintain at least a $2.5\ lb/in$ stiffness ($C_h = 0.4\ in/lb$) during teleoperation. In this typical situation, the manipulator/environment stiffness is much higher than the operator's-hand/hand-controller stiffness ($K_{me} C_h$=10), and from (2) the maximum force reflection gain G_{fr} is limited to only 1/10 for the unity position scaling factor (G_{ps}=1). Our foregoing analysis clearly indicates that the poor force reflection is not due to a poor implementation of the specific systems, but rather inherent to the existing conventional force-reflection system with dissimilar master-slave arms. A good direction to increase the force reflection gain is to make the robot more compliant by employing compliant control.

Shared compliance control has been implemented recently [9] by low pass filtering the outputs of the force/torque sensor mounted on the base of the robot and using these signals to alter the human operator's position/orientation command (Fig. 4). This low-pass-filtered force/torque feedback has an effect of giving the robot hand behavior similar to a damped spring (in each of the task space dimensions) in series with the stiff, position-controlled, robot manipulator. An approximate mechanical equivalent of the above implementation consists of a spring connected in parallel with a damper. It can be shown that the compliance control force feedback gain G_{cc} is approximately the new compliance value of the manipulator system in Fig. 4.

We now consider a simple combination of FR with SCC as shown in Fig. 5. This combination results in a system having two feedback loops; the inner compliance control loop residing in the robot side, and the outer force reflection loop with the operator in the loop. At first glance, one might think from (2) that the simple combination of SCC and FR of Fig. 5 should increase the force reflection gain markedly, since the inner compliance control loop makes the manipulator/environment stiffness K_{me} very low, approximately $1/G_{cc}$. Experimental testings however revealed that this simple combination increases the maximum force reflection gain only slightly. This can be understood by noting that the compliant control has a low pass filter whose bandwidth is lower than the manipulator bandwidth. As the frequency increases above the low pass filter bandwidth, the effect of the inner compliant control loop diminishes resulting in the original model of Fig. 1, and thus in this scheme SCC does not contribute much to improve the force reflection gain.

An alternate way of providing FR is to utilize the position error between the commanded and the actual position of the robot arm. Namely, we can have force reflection proportional to the position error Δx, namely $f_{hc} = G_{ps} \Delta x$. Although this position-error-based force reflection technique has been widely used in replica master-slave arms as a standard approach to achieve the unity force reflection gain, its

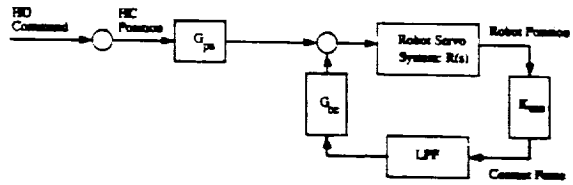


Fig. 4. Shared compliance control implementation with low-pass-filtered force/torque feedback.

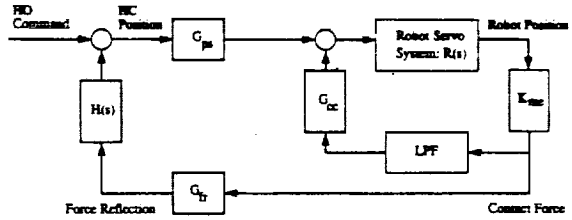


Fig. 5. A simple combination of force reflection with shared compliance control. This scheme does not increase the force reflection gain noticeably.

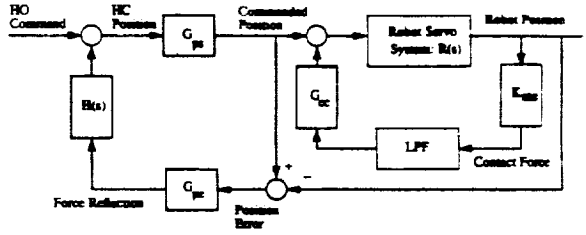


Fig. 6. Position-error-based force reflection with compliance control.

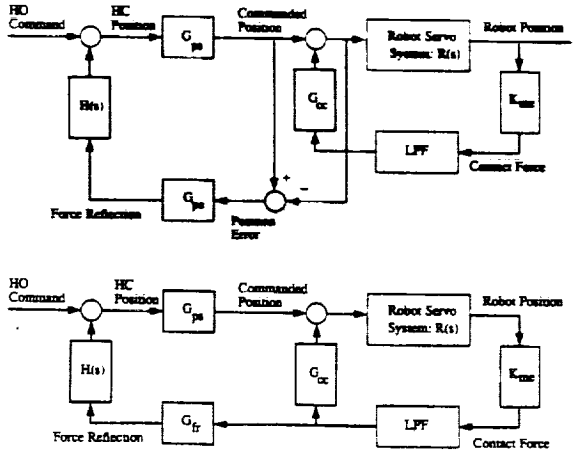


Fig. 7. A variation of the position-error-based force reflection with compliance control (a) and its equivalent conversion resulting in low-pass-filtered force reflection with compliance control (b).

implementation to dissimilar master-slave arms resulted in poor force reflection, since the slave arm is usually much stiffer than the operator's hand holding the hand controller (master arm). We have recently succeeded in developing a new scheme of force reflecting control that enables the system to have a sufficiently high force reflection gain (up to 2 or 3) for dissimilar master-slave arms by combining the position-error-based force reflection with compliance control (Fig. 6). Compliance control is essential to achieve high force reflection gain. In this scheme the force reflection gain is given by $G_{pe}G_{cc}$, since the contact force f_{rh} at the robot hand deflects the hand by $\Delta x = G_{cc}\, f_{rh}$, and the drive force of the force reflecting hand controller is then related to the robot contact force by $f_{hc} = G_{pe}\Delta x = G_{pe}G_{cc}\, f_{rh}$. It is interesting to observe that in this scheme the force/torque sensor outputs are not directly used for force reflection. Instead, the force/torque sensor outputs are used for robot compliance control, while the position/orientation errors which are generated in proportion to robot compliances are used for force reflection.

A variation of the position-error-based force reflection has eventually led to an alternate scheme that also enabled the system to have high force reflection. By noting that the robot servo system cartesian-space transfer function for each cartesian axis is close to 1 for low frequencies ($R(0)=1$), the control scheme of Fig. 6 is slightly changed as shown in Fig. 7a, which can then be equivalently converted to Fig. 7b with $G_{fr} = G_{pe}G_{cc}$. This resulted in another new scheme of force reflecting control. In this scheme, low-pass-filtered contact forces, instead of pure uncompensated forces, are fed back to the operator. Note that a simple idea of combining pure force reflection and compliance control of Fig. 5 did not allow high force reflection, while this new scheme enables the system to have high force reflection (up to 2 or 3) by using low-pass-filtered force reflection, instead of uncompensated pure constant gain force reflection, is used in combination with compliance control. The above two newly developed schemes --- position-error-based force reflection with compliance and low-pass-filtered force reflection with compliance --- appear to be similar in characteristics and performance. In both schemes, high force-reflection is achieved only with a limited bandwidth that is the same bandwidth imposed by the low pass filter of the compliance control compensator. An interesting feature observed in the position-error-based force reflection is that the operator feels artificial force when the operator moves the hand controller faster than the actual robot motion.

Compliance, Force Reflection, and Stability Measurements

In order to characterize the force reflection and compliance behavior of the system, the force-input/digital-output characteristic of the force/torque sensor [3] and the digital-input/force-output characteristic of the force reflecting hand controller [1] were roughly measured manually by using a force gage. Measurements indicate that the force/torque sensor reading is fairly linear up to $\pm$ 10 *lb* for the x, y, z translations (Fig. 8, upper panel) and $\pm$12 *lb·in* for the roll,

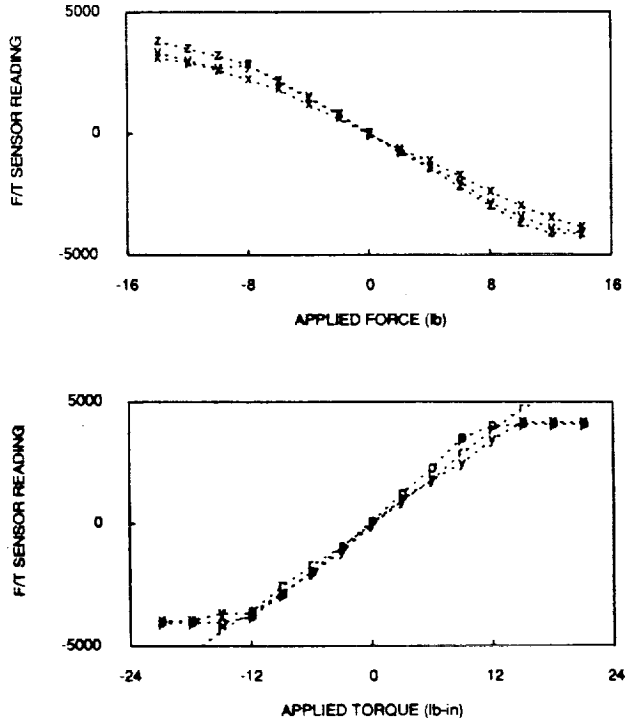


Fig. 8. Digital readout vs. applied input force/torque measurements of the force/torque sensor for x, y, z translations (upper) and for roll, pitch, yaw rotations (lower).

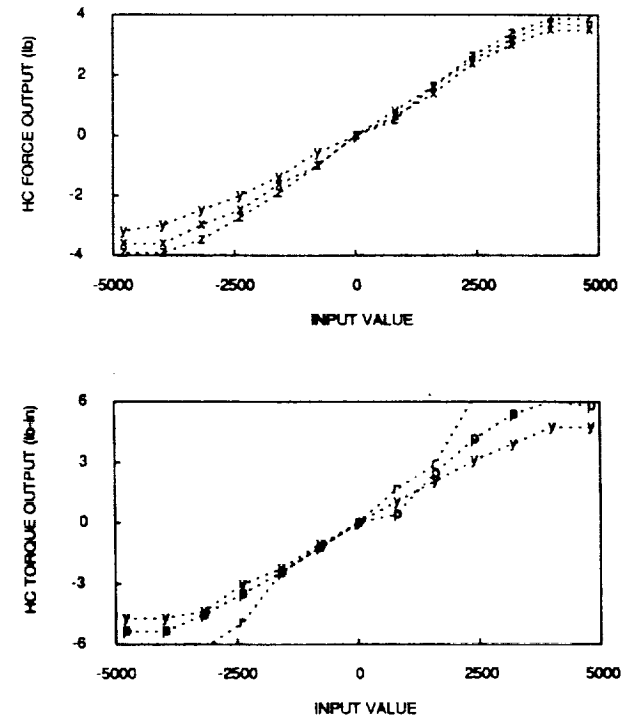


Fig. 9. Force/torque output vs. digital input measurements of the force reflecting hand controller for x, y, z translations (upper) and for roll, pitch, yaw rotations (lower).

pitch, yaw rotations (Fig. 8, lower). The force/torque drive behavior of the force reflecting hand controller is fairly linear up to about ±4 *lb* (Fig. 9, upper) for translations and about ±4 *lb·in* for rotations (Fig. 9, lower).

Compliance measurements (robot hand deflection vs. applied force) of SCC of Fig. 4 were plotted in Fig. 10 for four compliance feedback gains, G_{cc} = 1/16, 1/8, 1/4, and 1/2 *in/lb*. The plots show that the new compliance value of the robot hand is approximately equal to the compliance compensator feedback gain G_{cc}. The measured compliance data also show excellent linearity in the robot work volume. In the SCC implementation, a low pass filter is used to add damping to stabilize the system. A larger compliance means a higher compliance feedback gain (G_{cc}), which requires a lower bandwidth of the low pass filter with a more sluggish compliant response. The maximum bandwidths of the low pass filter for given desired compliance values were measured and plotted in Fig. 11. The maximum bandwidth of the low pass filter is about 3.4 *Hz* for the compliance value of G_{cc} = 1/16 *in/lb* (16 *lb/in* stiffness), 1.6 *Hz* for 1/8 *in/lb*, 0.8 *Hz* for 1/4 *in/lb*, and 0.4 *Hz* for 1/2 *in/lb*. In the above measurements, compliance compensators were added only along translational axes not about rotational axes. When both were enabled, the maximum bandwidth values were reduced further approximately to a half. A more detailed stability analysis can be found in [6].

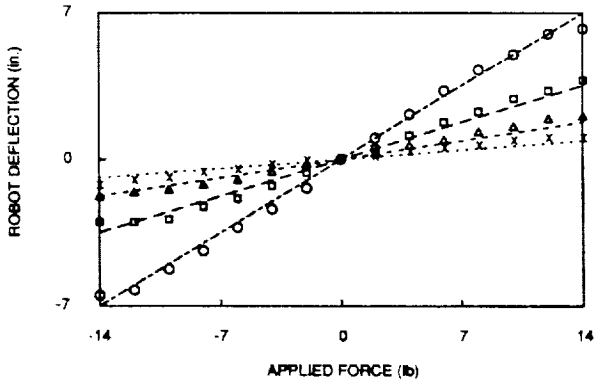


Fig. 10. Compliance measurements of the shared compliance control: robot hand position deflection vs. applied force to the robot hand for four compliance compensator feedback gains of G_{cc} = 1/16 (x), 1/8 (triangle), 1/4 (square) and 1/2 (circle) *in/lb*.

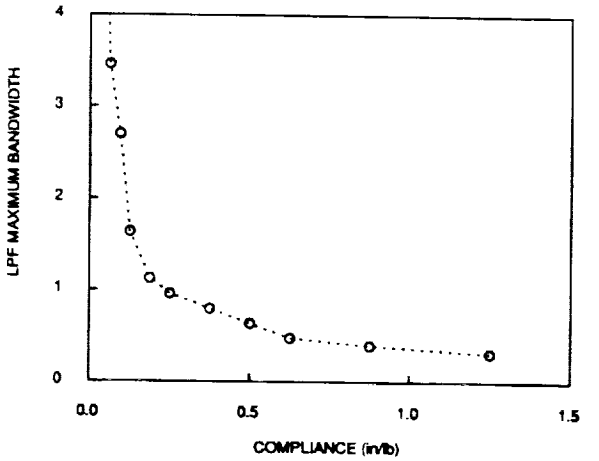


Fig. 11. Maximum bandwidth of the low pass filter vs. compliance value (compliance compensator feedback gain) measurements for the shared compliance control of Fig. 4.

The force reflection behaviors of the position-error-based force reflection scheme of Fig. 6 were measured (Fig. 12) for the three force reflection gains of 1/4 (G_{cc} = 1/16 *in/lb*), 1/2 (G_{cc} = 1/8 *in/lb*), and 1 (G_{cc} = 1/4 *in/lb*) with a fixed position error gain of G_{pe} = 4 *lb/in*. Note that the force reflection gain in this scheme is given by $G_{pe}G_{cc}$. In Fig. 12, all three curves saturate at about 4 *lb* drive force, since the maximum drive force of the force reflecting hand controller is limited to about 4 *lb* as shown in Fig. 9. This limited drive force is probably a good feature since excessive force in the hand controller causes rapid operator fatigue.

Fig. 13 is a plot showing the maximum bandwidth vs. the force reflection gain for the position-error-based force reflection with three different compliance values of the compliance compensator (G_{cc} = 1/16, 1/8, 1/4 *in/lb*). For a given compliance value, both the bandwidth and the force reflection gain are limited. It is interesting to observe that an abrupt oscillation occurs as soon as the force reflection gain exceeds a certain maximum value. In Fig. 13, the maximum bandwidths for the compensator compliance values of 1/16, 1/8, 1/4 *in/lb* are 3.4 *Hz*, 1.6 *Hz*, 0.8 *Hz*, respectively, and the maximum force reflection gains for the same compliance values are 0.375, 0.75, 1.5, respectively. These data indicate that the maximum bandwidth is inversely proportional to the compliance value, while the maximum force reflection gain is proportional to the compliance value. The maximum bandwidths are limited by the stability boundary of the compliance control feedback loop as described earlier (Fig. 11). The maximum force reflection gains are somewhat higher than expected from (2), and a more careful stability analysis is in progress.

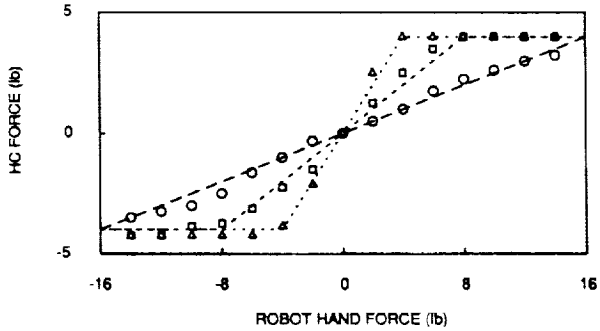


Fig. 12. Force reflection characteristics of the position-error-based force reflection combined with compliance control for the force reflection gains of 1/4 (circle), 1/2 (square), and 1 (triangle).

The maximum force reflection gains of the position-error-based force reflection with four different position scale factors (G_{ps} = 1/4, 1/2, 3/4, 1) were measured and plotted in Fig. 14 for four different compliance values of the compliance compensator (G_{cc} = 0, 1/16, 1/8, 1/4 *in/lb*). The maximum force reflection gain is inversely proportional to the position scale factor G_{ps}, which can be easily conjectured from (2). We can observe in Fig. 14 that the maximum force reflection gains are approximately doubled when the position scale factor is doubled, for example, from 1/2 to 1. The position-error-based force reflection is possible without compliance control (G_{cc}=0) as seen in Fig. 14, but the maximum force reflection gain is limited to about 1/10 for the unity position scale factor.

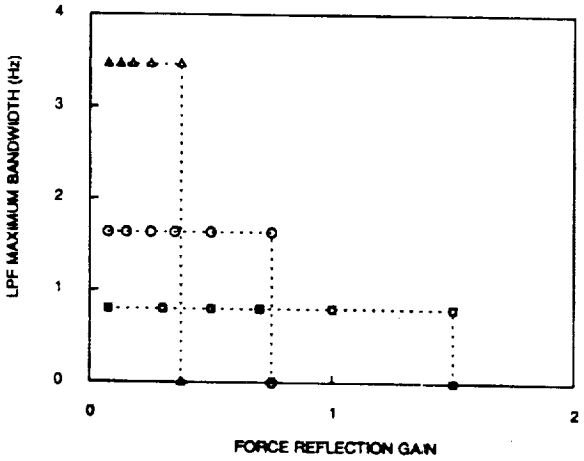


Fig. 13. Maximum bandwidth of the low pass filter vs. force reflection gain measurements of the position-error-based force reflection with compliance control for three compliance values of 1/16 (triangle), 1/8 (circle), and 1/4 (square) *in/lb*.

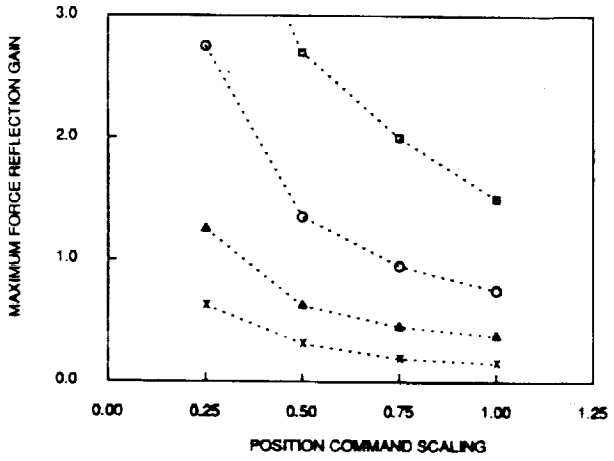


Fig. 14. Maximum force reflection gain vs. position scale factor measurements of the position-error-based force reflection with compliance control for four compliance values of 0 (x), 1/16 (triangle), 1/8 (circle), and 1/4 (square) *in/lb*.

Peg-in-Hole Experiments with Different Operating Modes

Peg-in-hole tasks were performed with eight different operating modes to evaluate the position-error-based force reflection in comparison with other operating modes. A 7"x7" peg-in-hole task module mounted on the 21"x21" task board [5] was used for the peg-in-hole task. The peg-in-hole task module has 9 holes arranged in a square matrix. In our experiments, only one hole with 10 mil clearance and no chamfer was used. The peg was 4.75" in length and 0.998" in diameter. The peg-in-hole task consisted of the following steps: i) the peg is initially located at about 2 inches in front of the designated hole of the peg-in-hole task module, ii) move the peg to the designated hole, iii) insert the peg into the hole completely, iv) extract the peg. In our Advanced Teleoperation setup, the hand controller of the master side was installed in the control station room separate from the PUMA arm of the slave side. Three television camera views of the task board and robots were provided in the control station: top, upper left, and upper right views of the task environment. The focus and zoom settings were fixed throughout the experiments. During the experiments, force/torque data of the robot hand were recorded to a hard disk at 100 Hz sampling rate through a parallel I/O port of an IBM computer.

The eight operating modes tested are: (mode 1) low-pass-filtered FR combined with SCC with the FR gain = 1/2, (mode 2) position-error-based FR combined with SCC with the FR gain = 1/2, (mode 3) low-pass-filtered FR combined with SCC with the FR gain = 1/4, (mode 4) SCC only, (mode 5) damper only control with no active compliance, (mode 6) uncompensated pure FR with the FR gain = 1/10, (mode 7) pure position control without FR or SCC, and (mode 8) rate control with SCC. For all position control modes of 1 through 7, the position scale factor is fixed to G_{ps}=1/2. The stiffness values (inverse of the compliance values) used for SCC were 6.7 *lb/in* (80.0 *lb/ft*) for cartesian translations and 2.8 *lb·in/deg* (13.4 *lb·ft/rad*) for cartesian rotations. The low pass filter bandwidths were 0.63 *Hz* for translations and 0.47 *Hz* for rotations. For simplicity, the same compliance and bandwidth values were used for all three cartesian position axes, and so were for all three orientation axes, and no serious attempt was made to find the optimal parameter values.

In the experiments, test operators performed the peg-in-hole task three times each with the 8 operating modes in random order (24 tasks in total). Three test operators participated in the experiments. All operators first trained themselves until they could complete the peg-in-hole task comfortably for all operating modes. Then, each operator performed one complete set of the experiment of 24 peg-in-hole tasks as a practice run. Thereafter, actual experiment was performed for experimental data collection.

Task completion times and cumulative contact forces were computed from the contact force/torque data recorded during the experiment and the means and standard deviations of the three test operators' data are plotted in Fig. 15. From Fig. 15, we can observe that completion times are similar for all position control modes, but contact forces are

greatly reduced with the use of SCC and/or FR. Performance with position control (modes 1 through 7) is superior to that with rate control. The best task performances resulted with our newly developed schemes --- position-error-based FR with SCC and low-pass-filtered FR with SCC. Both schemes combine FR and SCC, and enable high force reflection with limited bandwidths. Due to limited bandwidth, operators felt force reflection sluggishness during the peg-in-hole task execution. Some operators felt more comfortable with a reduced force reflection gain of 1/4 compared to 1/2, although the task performance was better with the force reflection gain of 1/2 in terms of cumulative contact force as shown in Fig. 15. Performance with SCC only or damper only was superior to that with uncompensated pure force reflection (force reflection gain = 1/10) as seen in Fig. 15, which agree with previous experiments [7]. Low-pass-filtered FR alone without SCC was marginally operational, requiring the operator to maintain a very firm grasp during the peg-in-hole task performance, and thus was not included in our experiment.

Recently more thorough experiments with a screw insertion/removal task [4] were performed with seven test operators to compare various control modes. Again the newly developed position-error-based force reflection combined with compliance control resulted in the best task performance among all control modes tested.

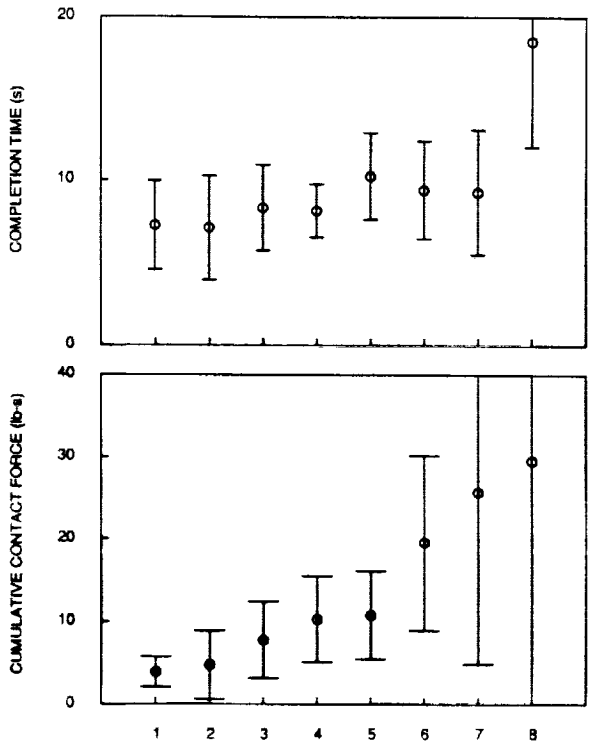


Fig. 15. Completion time (a) and cumulative contact force (b) plots for the peg-in-hole task (1-inch-diameter peg with 10-mil clearance) with 8 different operating modes. Means (circles) and standard deviations of three test operators' data are plotted. Mode 1: low-pass-filtered force reflection (FR gain = 1/2) with compliance, Mode 2: position-error-based force reflection (FR gain = 1/2) with compliance, Mode 3: low-pass-filtered force reflection (FR gain = 1/4) with compliance, Mode 4: shared compliance control only (compliance and damper), Mode 5: damper only, Mode 6: pure uncompensated force reflection only (FR gain = 1/10), Mode 7: pure position control, and Mode 8: rate control with compliance.

Conclusion

A new scheme of force reflecting control --- position-error-based force reflection combined with compliance control --- has been developed for dissimilar master-slave arms. This new scheme has enabled the system to have high force reflection gain (up to 2 or 3), which was not possible with a conventional scheme when the slave arm is much stiffer than the master arm. The experimental results with a peg-in-hole task indicate that the newly developed position-error-based force reflection combined with compliance control resulted in best task performance.

Acknowledgment

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration

References

[1] A. K. Bejczy and J. K. Salisbury, "Kinesthetic Coupling Between Operator and Remote Manipulator," Proceedings: ASME International Computer Technology Conference, vol. 1, pp. 197-211, San Francisco, CA, Aug. 1980.

[2] A. K. Bejczy, Z. Szakaly, and W. S. Kim, "A Laboratory Breadboard System for Dual-Arm Teleoperation," Third Annual Workshop on Space Operations Automation and Robotics (SOAR '89), pp. 649-660, NASA Johnson Space Center, TX, July, 1989.

[3] A.K. Bejczy, Z. Szakaly, and T. Ohm, "Impact of End Effector Technology on Telemanipulation Performance," Third Int. Workshop on Space Operations Automation and Robotics (SOAR '89), pp. 429-440, NASA Johnson Space Center, Houston, TX, July 1989.

[4] H. Das, H. Zak, W. S. Kim, A. K. Bejczy, and P. Schenker, "Performance Experiments with Alternative Advanced Teleoperator Control Modes for a Simulated Solar Max Satellite Repair," Fifth Annual Workshop on Space Operations, Applications, and Research Symposium (SOAR '91), NASA Johnson Space Center, TX, July, 1991.

[5] B. Hannaford, L. Wood, B. Guggisberg, D. McAffee, H. Zak, "Performance Evaluation of a Six-Axis Generalized Force-Reflecting Teleoperator," JPL Publication 89-18, June 1989.

[6] W. S. Kim, "Shared Compliant Control: a Stability Analysis and Experiments," IEEE Conf. on Systems, Man, and Cybernetics, pp. 620-623, Los Angeles, CA, Nov. 1990.

[7] W. S. Kim, P. G. Backes, S. Hayati, and E. Bokor, "Orbital Replacement Unit Changeout Experiments with a Telerobot Testbed System," IEEE Int. Conf. on Robotics and Automation, pp. 2026-2031, Sacramento, CA, April 1991.

[8] W. S. Kim and A. K. Bejczy, "A Stability Analysis of Shared Compliance Control," Japan-USA Symposium on Flexible Automation, pp. 567-572, Kyoto, Japan, July 1990.

[9] W. S. Kim, B. Hannaford, and A. K. Bejczy, "Shared Compliance Control for Time-Delayed Telemanipulation," First Int. Symposium on Measurement and Control in Robotics, NASA Johnson Space Center, Houston, TX, June 1990.

Fault Detection and Fault Tolerance in Robotics

Monica Visinsky, Ian D. Walker, and Joseph R. Cavallaro
Department of Electrical & Computer Engineering
Rice University
Houston, TX 77251-1892

Abstract

Robots are used in inaccessible or hazardous environments in order to alleviate some of the time, cost and risk involved in preparing men to endure these conditions. In order to perform their expected tasks, the robots are often quite complex, thus increasing their potential for failures. If men must be sent into these environments to repair each component failure in the robot, the advantages of using the robot are quickly lost. Fault tolerant robots are needed which can effectively cope with failures and continue their tasks until repairs can be realistically scheduled. Before fault tolerant capabilities can be created, methods of detecting and pinpointing failures must be perfected. This paper develops a basic fault tree analysis of a robot in order to obtain a better understanding of where failures can occur and how they contribute to other failures in the robot. The resulting failure flow chart can also be used to analyze the resiliency of the robot in the presence of specific faults. By simulating robot failures and fault detection schemes, the problems involved in detecting failures for robots are explored in more depth. Future work will extend the analyses done in this paper to enhance Trick, a robotic simulation testbed, with fault tolerant capabilities in an expert system package.

1 Introduction

In hazardous environments or environments which are not readily accessible to man, robots must be able to efficiently adapt to failures in both software and hardware in order to continue working until the problem can be realistically repaired. Before a robot can try to cope with a failure, however, it must first be able to detect and pinpoint the problem. This paper develops a basic fault tree analysis of robots in order to obtain a better understanding of where failures can occur and how they contribute to other failures or limitations in each robot. The resulting flow chart style picture of failures in a robot can also be used to analyze the resiliency of the robot in the presence of specific faults. Once a failure has been detected, the robot can reorganize its view of its internal structure in such a way as to hide or isolate the fault so the robot can continue working. The focus of this research is on finding real-time fault detection and fault tolerance methods which maintain as much of the robot's functionality as possible while not requiring that redundant or extra parts be added to the robot.

1.1 Previous Work

1.1.1 Redundancy Based Fault Tolerance

Previous work on fault tolerance in robotics has concentrated on dealing with faults in one specific part of the robot (mechanical failure in the motor, kinematic joint failure, etc.) with only token thought going to the more critical, systemwide effect of the failures. Relatively little focus has been given to the question of how to detect failures in robots. Previous research tends to concentrate on fault tolerance algorithms, especially those schemes which rely on duplicating parts such as joint motors [15,19] for their fault tolerant abilities. These redundancy based schemes are similar to several computer fault tolerant algorithms which are also based on redundancy of parts. One common computer fault tolerant algorithm is Triple Modular Redundancy (TMR) in which three processors all work on the same problem and compare their results. If one of the processors is faulty and its result does not agree with the results of the other two processors, the faulty processor is voted out of the final decision and the correct result is passed on to the rest of the system. This fault tolerance scheme fails, however, if more than one of the processors is faulty. Duplication of physical parts provides a backup in case the element performing the work fails. Redundancy of parts can also provide a useful means of checking to see if a component is in error.

For the equivalent redundancy based robot fault tolerant algorithms, two motors have been placed in each robot joint to provide a backup in case one motor stalls, runs away, or begins free-spinning. The fault tolerant advantages of redundancy have also led to adding extra parallel structures, such as seven legs when only six are needed, in order to allow many different configurations in the presence of a failure. Previous work by Tesar, et al at UT, Austin [15] and independently by Wu [19] with Lockheed at Johnson Space Center have explored the aforementioned method of duplicating motors. Two motors in a joint work together so as to provide one output velocity for the joint. When one of the motors breaks, the other one takes over the faulty motor's functions. The faulty motor must be isolated from the system or the second motor must be able to adjust its output to account for any transients introduced to the system by the failed motor. If the robot is performing a time-critical or delicate task, fault tolerance must allow the robot to get a run-away motor under control quickly before any damage to the environment or the robot occurs.

1.1.2 Structure-Independent Fault Tolerance

Many useful robots have already been created. In order to provide fault tolerance for these robots without redesigning them, algorithms need to be developed that will utilize the advantages of the existing structure and not require the addition of extra motors, sensors, or other components to the robot [16]. These algorithms should be easily adaptable to most robots regardless of the robot structure.

To avoid adding redundant parts for fault tolerance in computers, algorithms have been developed which reconfigure the data or code in a computer system among the working parts after one component has failed. Some computer fault tolerant systems handle a fault by allowing a graceful degradation in functionality or speed. The literature speaks of time redundancy [4] in which a computational cycle is lengthened so a fault-free part (or parts) will have enough time to handle the tasks of a faulty component. Other systems use set-switching or processor-switching schemes [4] for reconfiguration. In processor-switching, fault-free components can be collected to form a basic subpart of the configuration, such as a row of an array, until the full configuration is achieved. This method may, however, require many extra interconnections between components. In software, check bits and error correction codes help insure that data is successfully transmitted in the system and allow a reconstruction of the original data if a transmission line is faulty.

For robotics, however, little work has been done in developing algorithms for accommodating a failure using only the available physical parts. Many robots exist which do not have redundant motors or extensive sensors in the joints. Duplicating motors increases the size of the robot, the cost involved in building it, and the weight and inertia which affect the robot controller. It would thus be cost effective to find fault tolerance schemes that do not rely on a specific robotic architecture to continue working but reorganize the robotic algorithms in the controller or utilize the self-motion capability of robots with redundant joints. In order to develop these schemes, the advantages and capabilities of a general robot architecture must be researched.

Maciejewski at Purdue University has quantified the effect of joint failure on the remaining dexterity of a kinematically redundant manipulator [10]. He calculates an optimal configuration of redundant arms to maximize the fault tolerance while minimizing the degradation of the system in the event of a failure. His method currently only provides fault tolerance if the robot is near this initial configuration and can try and arrange its joints to mimic the fault safe configuration as close as possible. Robot controllers may further attempt to keep the robot arm in a configuration where the joints are arranged to stay away from any possible singularities or uncomfortable positions in case a joint fails during the operation. These studies do not rely on adding extra motors or other components to the robot, but they also do not explore what the robot controller must do in order to utilize the remaining dexterity to continue its tasks.

This paper considers and analyzes systemwide failures (electromechanical, computer software/hardware, etc.) and their inter-relationship via fault trees. We focus on developing fault detection and fault tolerance schemes using only the components normally available to the robot. Previous work in fault tolerance forms a subset of our analysis, and our structure has the additional advantage of allowing the best results from more specific fault schemes to be embedded into our tree analysis.

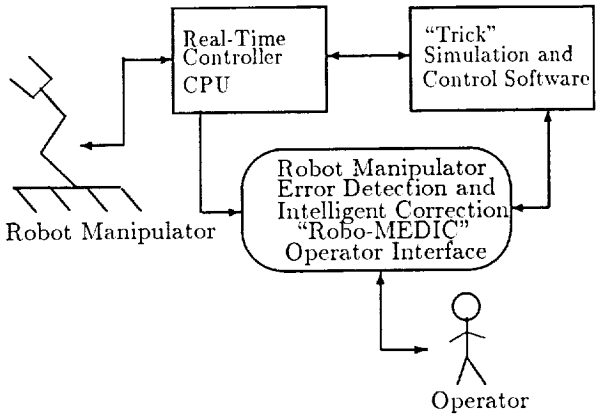


Figure 1: Robo-MEDIC Fault Tolerance Environment.

1.2 Riceobot and Robo-MEDIC

This paper begins by specifically analyzing the fault trees of the Rice University robot, the Riceobot, but the results apply to most robots. The fault tolerant algorithms developed from this analysis will be embedded into the CLIPS expert system environment [6]. This NASA-developed public domain software package is commonly used by government agencies and is running on our computer systems.

The resulting expert system package, Robo-MEDIC (Robot Manipulator Error Detection and Intelligent Correction) will provide diagnostic assistance to the operator and will interface with the control computer of the robot as shown in Figure 1. Robo-MEDIC will be able to use the fault trees as a flow chart of failures. Nodes in the trees will have some fault tolerant action associated with them that will allow the robot to take advantage of inherent backup or alternate paths charted by the fault tree. By maneuvering around the trees, Robo-MEDIC will perform fault tolerant recovery actions as a sequence of these smaller, simpler actions.

1.3 Fault Detection Simulator and Trick

In addition to the fault tree analysis, we are examining failures and testing fault detection schemes using a simulation of a generic four link, planar robot. We will be integrating the concepts derived from the simulator into Trick [1], a robotics software testbed developed at NASA Johnson Space Center by Leslie J. Quiocho and Robert Bailey.

The Trick software package already contains information to model the seven-joint Robotic Research Arms, the Space Shuttle RMS, and the full Riceobot with base and two arms. Data modules provided by Trick allow the user to build customized robots with various types of sensors, joints, and links. Our research is expanding the capabilities of the software to model fault detection and tolerance algorithms. The flexibility of the software allows the failure analysis developed in this paper to be extended to a variety of different robots.

Table I: Fault Tree Analysis Symbols

Symbol	Function
AND gate	All inputs required to produce output event.
OR gate	Any one input event causes the output event.
Rectangle	A malfunction which results from a combination of fault events through logic gates.
Diamond 	A fault event for which the causes are left undeveloped.
Circle 	A basic fault event. This includes component failures whose frequency and failure mode are known.
Triangle 	A suppressed tree. The tree is detailed in another figure.

2 Robotic Fault Tree Analysis

2.1 Analysis Technique

Fault Tree Analysis (FTA) is a deductive method in which failure paths are identified by using a fault tree drawing or graphical representation of the flow of fault events [2]. FTA is a well-known analysis technique often used in industry for computer control systems and large industrial plants. Each event in the tree is a component failure, an external disturbance, or a system operation. The top event is the undesired event being analyzed and, in this research, is the failure of the entire robot. The events are connected by logic symbols to create a logical tree of failures. Some of the basic symbols are explained in Table I.

The explanation of the FTA technique in [2] promotes a top down development of the fault tree. The top event is broken down into primary events that can, through some logical combination, cause the failure at the top. This process is repeated to deeper levels until a basic event or an undeveloped event is reached. Some conditions or causes may be left undeveloped if the probability that they will occur is small enough to be ignored.

2.2 Failure Propagation/Probability Analysis

The information available in the fault trees may be enhanced by a quantitative analysis of the failures. Failure rates are assigned to each input event and propagated up the tree based on the rules of the connecting logic gates. The output of an OR gate is the sum of the inputs. The resulting probability of the combined input events is greater than the probability of an individual input event. The output of an AND gate is the product of its inputs. The probability of all the events occurring is less than the probability of any one occurring. The AND gate represents a redundant measurement or capability and is more desirable in the tree since the probability of a failure decreases through the combination of lower level events.

A Markov or semi-Markov model approach to probability analysis can also be developed based on the PAWS/STEM [3] and CARE III [14] reliability analysis packages. These packages can analyze simpler fault trees as well as Markov chains, but they are not necessarily optimized to handle the simpler structures [11]. These analysis tools were also not designed for robotics and thus would not take advantage of some of the commonality within robot structures. We will include some of the advantageous aspects of using Markov models in our CLIPS-based expert system, Robo-MEDIC.

A quantitative analysis provides a measure of the overall chance of a complete failure for each robot. The structure provided by the fault trees organizes the probabilities appropriately for the robot system and provides a simple map of how the probabilities relate to each other. Using the trees, robots of significantly different origin and structure can be compared for fault tolerant abilities and survivability. The integrated Robo-MEDIC expert system will provide diagnostic capabilities by using the fault trees and will alert the operator of an impending failure. It can be used for off-line comparisons of robots or for suggesting possible corrective actions to an operator and the low-level robot controller during real operations.

2.3 Fault Tree Pruning

A suggested drawback of FTA is that there is no way to ensure that all the causes of a failure have been evaluated [2]. The designer tends to identify the important or most obvious events that would cause a given failure. However, the events that are not modeled normally have a low probability of occurring and can be ignored or treated as a basic event without overly biasing the analysis.

Several failures may also be interconnected creating lateral branches or cycles in the fault tree. In some robots, one motion at a joint may be coupled with another motion such that failure of either motion causes the failure of the other. It is also difficult to determine the relationships between some failures. For example, the failure of all the internal feedback sensors at the elbow joint of a robot may make the robot blind to the elbow's position. The elbow has not actually failed, but the robot is unable to detect the results of any commands sent to the elbow. Thus, the sensor malfunction does not contribute to an elbow failure specifically but may cause a failure of the entire robot. Relationships like these make the tree complex and difficult to analyze. These problems can be overcome by working to simplify the tree. In the case of the coupled motions, the two failures can be considered as one with twice as likely a probability of occurring.

2.4 Riceobot Fault Trees

To provide a foundation for the analysis of general robots, we have chosen to analyze the arm of the Rice University Riceobot. The arm has eight degrees of freedom: three motions in the shoulder (z translation, pitch, and yaw), two motions in the elbow (roll and pitch), and three motions in the wrist (roll, pitch, and yaw). The results obtained from the Riceobot apply to most general robots especially since the Riceobot has a wide variety of commonly used link, joint, and motor arrangements.

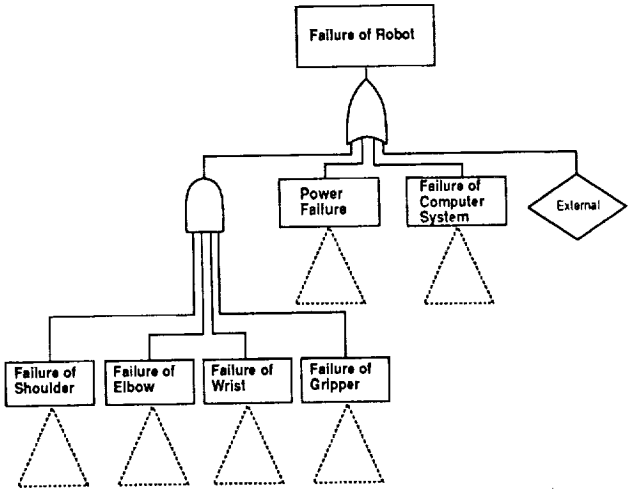


Figure 2: Top Level Fault Tree for Entire Riceobot.

Overall Robot Failure

Several fault trees have been developed and a few are reproduced in the following pages. The top event is obviously the failure of the entire robot (Figure 2). The primary causes of a robot failure are power failure, computer system failure, or a combination of failures of the joints. If the robot is fault tolerant, it can withstand the failure of several joints. By stablizing the faulty motion or joint in some manner (such as locking the joint), it is possible that the other motions can still provide some functional capability to the robot. This ability results in the AND gate combining the joint failures in Figure 2 and decreases the probability of a failure of the robot.

Joint Failures

The Riceobot has two directly driven motions: the shoulder z-direction motion and the wrist roll motion (Figure 3). The fault trees for these motions are quite simple since only the failure of the motor plays an important role in the failure of the motion. The other motions of the Riceobot depend on some form of gear-train assembly to allow the spatially separated motor to drive the joint. Failure of the gear-train can be caused by basic events as simple as a loosening of the chain or cable.

Motor and Sensor Failures

The probability of a motor failure is dependent on the type of motor used. The Riceobot contains both brushless DC and stepper motors. Each motor also has a gear box which may fail due to gear slippage or wear. A power failure affects all motors as well as any other electrically driven parts in the robot, but each motor could lose power separately if its specific power cables break. A motor failure could conceivably also be the cause of a sensor failure when sensors are mounted on the shafts of the motors. Sensors are also affected by incorrect calibration and external noise or vibrations (see Figure 4).

Computer System Failures

The computer system of the Riceobot consists of three main parts: (1) amplifiers which read from the optical encoders and drive the motors, (2) servo control chips which store information about the different motors and convert the desired angles into currents for each motor, and (3) an on-board host computer which is programmed in C and computes the desired angles for the desired motions (Figure 5). These three parts each contain at least one board filled with TTL chips, capacitors, power transistors, resistors, and other analog and digital circuit components. A failure of any one of these parts may not cause a

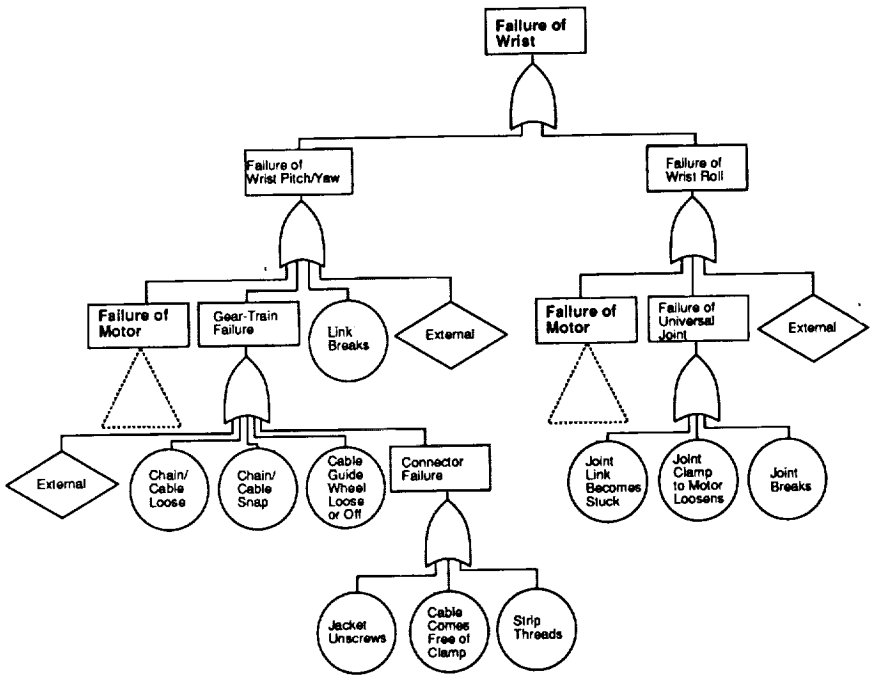


Figure 3: Sub-Level Fault Tree for Wrist System.

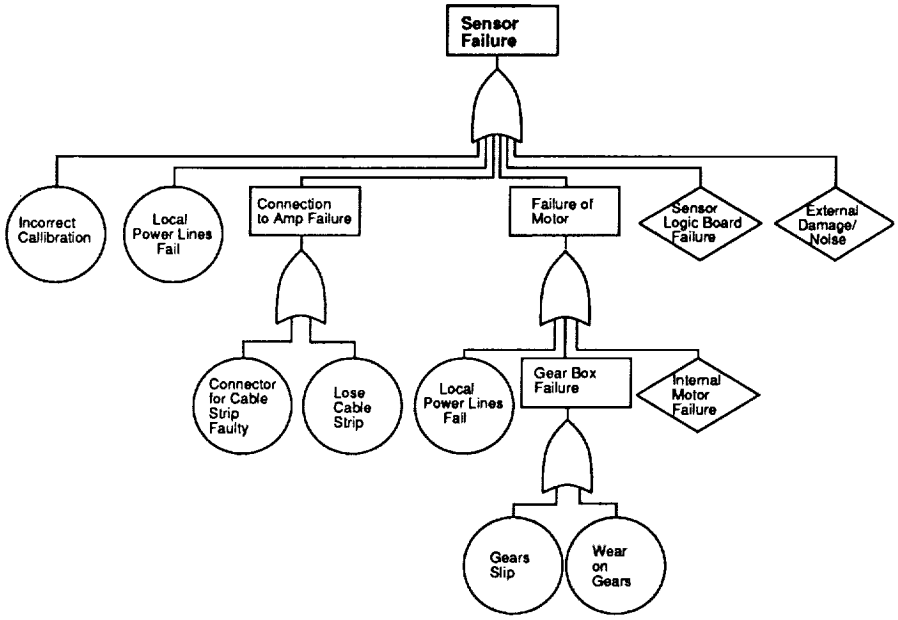


Figure 4: Sub-Level Fault Tree for Sensor System.

failure of the entire board; but if a board did fail, the robot would be unable to function. The robot cannot withstand the failure of all the servo controllers or all the amplifiers because it would no longer be able to communicate with the joints.

Detecting a non-terminal failure in the computer system requires some form of testing circuitry or the ability to poll components to see if they are still alive. The IEEE standard 1149.1 Test Access Port may be incorporated into any VLSI chip on the boards and could be used for active testing [9]. Radiation hardened circuits [18] should also be used in the computer system. Correction code bits can be used to check data transfers and could identify a bus failure if the bits were consistently wrong.

2.5 Derived Riceobot Fault Detection

The qualitative analysis of these fault trees has proven useful in pointing out some of the limitations of the Riceobot in regards to fault detection and fault tolerance. With only one sensor at each joint, the Riceobot represents the worst case scenario for detecting sensor and joint failures. The only option available to the fault detection software is to compare the sensed angles with the calculated desired angles. After accounting for a predetermined threshold to mask any precision errors in the calculations or sensing equipment and possibly adjust for load effects, any difference between the sensed and desired values must be considered the result of a failure. The computer is, however, unable to differentiate between a sensor malfunction and an actual joint failure due, for example, to a frozen motor. The computer must therefore shut down the joint and proceed with fault tolerance schemes based on a new model of the robot with fewer possible motions.

Fault detection for the Riceobot could be improved using the vision system. With the computer calculation and the sensor reading, the additional joint angle information would help distinguish between a sensor error and a real joint failure. The Riceobot could then function in the presence of one sensor failure. Using the vision system for this task, however, increases the load on the image processing software and may hinder the system's ability to perform its normal vision tasks.

3 Robot Fault Detection

The fault trees give an indication of the interaction between failures in a system. The trees also provide a map of alternate paths for detecting faults or bypassing failures. In order to expand on this information and to show how modeling errors or other uncertainties affect fault detection, we need to simulate the robot and the fault detection algorithm. Because of the Riceobot's lack of sensors and the complexity needed for its fault detection algorithm, we are initially simulating fault detection using a computer modeled planar, four link robot [7]. The current program will need to be expanded extensively for the Riceobot and will be accomplished by implementing the fault detection routine in the CLIPS expert system as part of Robo-MEDIC.

The robot consists of four cylindrical links connected end-to-end. All joints are rotational and move in the same plane. A simulated optical encoder and tachometer were added for each joint. The fault tree for this robot is relatively simple (Figure 6). We have not included the possibility of link breakage or global power failure in the simulation. The motors are in essence direct drive with no gear-trains and fail only in a locked mode. These conditions pruned each joint subtree down from the complexity of the Riceobot trees to an easily simulated failure situation.

It is interesting to note that it is the fault detection software which allows the joint to survive in the presence of a single sensor failure thus creating the AND gate under each motor failure in the tree. If both sensors at a joint fail, the host computer is blind to that joint and the fault detection routine forces a motor failure to prevent the joint from moving too far with-

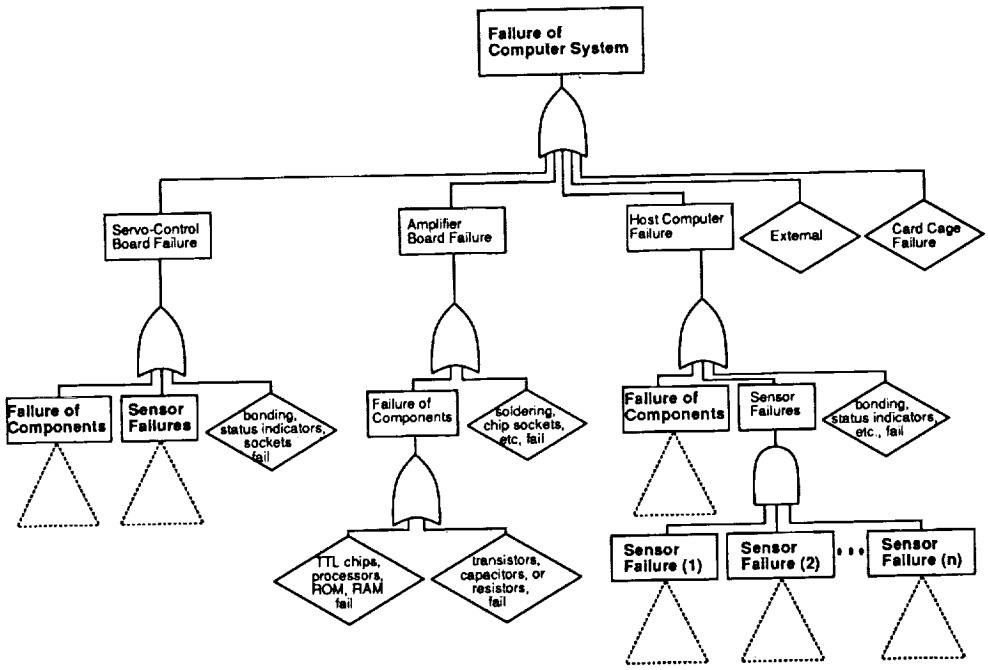


Figure 5: Sub-Level Fault Tree for Computer System.

out computer supervision. Thus, the fault detection algorithm makes the dual sensor failure subtree a cause of the motor failure events to protect a blind joint.

3.1 Fault Detection Simulator

The structure of the simulator is shown in Figure 7. The flow of information is from the simulated host computer through the robot and then the sensors to the fault detection program and finally back to the host computer. The host computer uses the desired angles computed by a planner routine and the estimated present position of the robot derived from the sensors to calculate the torque necessary to move each link to its desired destination. The controller is a standard proportional-derivative (PD) computed torque type controller. The robot routine then takes the calculated torques and determines the new position, velocity, and acceleration for each joint. The optical encoders estimate the positions by truncating the value of each angle based on each encoder's precision. The tachometers pass the velocities through a first order filter based on a predetermined motor lag time. The sensors are modules from the Trick simulation package and represent our initial efforts at integration with Trick. These estimates of the angles and velocities are passed into the fault detection procedure which checks for failures. Then, the procedure either passes to the host what it considers good estimates of the position, velocity, and acceleration or signals the motor of a joint with two bad sensors to shut down.

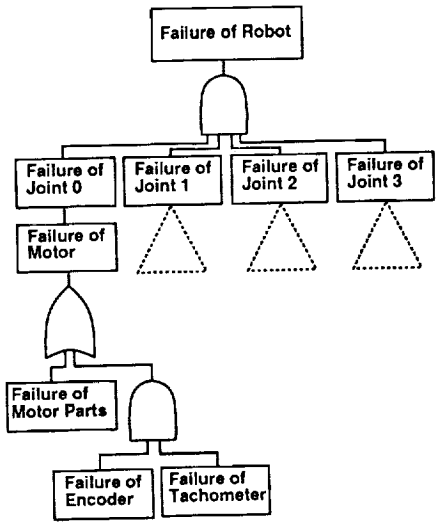


Figure 6: Four Link, Planar Robot Fault Tree.

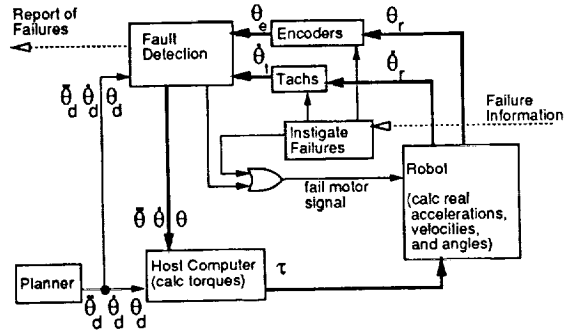


Figure 7: Fault Detection Simulator Flow Chart.

3.2 Host Computer Model

The simulated host computer uses the following dynamics equation as a model for the robot [7]:

$$\underline{\tau} = [M(\underline{\theta})]\underline{\ddot{\theta}} + \underline{N}(\underline{\theta}, \underline{\dot{\theta}}), \quad (1)$$

where $\underline{\tau}$ is the joint torque vector, $[M]$ is the inertia matrix, and $\underline{N}$ is the Coriolis and centrifugal torque vector. The $[M]$ matrix and $\underline{N}$ vector are computed based on the estimated angles from the sensors. Since the robot is planar, gravity is orthogonal to the plane of motion and there are no resultant gravity torques to consider. Friction is also neglected in this model.

The PD controller for this model becomes:

$$\underline{\tau} = [M(\underline{\theta})]\{\underline{\ddot{\theta}}_d + [K_P](\underline{\theta}_d - \underline{\theta}) + [K_D](\underline{\dot{\theta}}_d - \underline{\dot{\theta}})\} + \underline{N}(\underline{\theta}, \underline{\dot{\theta}}). \quad (2)$$

The matrices $[K_P]$ and $[K_D]$ are the position and derivative gains, respectively, and are used to control tracking and steady state errors by feedback control. For critical damping, the gains become:

$$[K_D] = 2\omega, \quad [K_P] = \omega^2, \quad (3)$$

where ω is the natural frequency input by the user. The natural frequency is typically set to 1 for most runs of the simulator.

3.3 Robot Model

The robot simulation takes the computed torque from the simulated host computer and determines the resulting four robot angle accelerations based on the equation:

$$\underline{\ddot{\theta}} = [\hat{M}]^{-1}\underline{\tau} - [\hat{M}]^{-1}(\underline{\hat{N}}). \quad (4)$$

Here, $[\hat{M}]$ and $\underline{\hat{N}}$ are the inertia matrix and Coriolis and centrifugal torque vector as before but are now based on the actual robot angles instead of the sensed angles. The matrices have also been injected with a small constant error to simulate modeling inaccuracies.

The joint angle, θ, and its first derivative are estimated by the equations:

$$\dot{\theta}_i = \dot{\theta}_{i-1} + (\Delta t)\ddot{\theta}_i, \quad (5)$$

$$\theta_i = \theta_{i-1} + (\Delta t)\dot{\theta}_i. \quad (6)$$

If a motor failure has occurred, $\ddot{\theta}$ and $\dot{\theta}$ are set to zero to simulate the effects of a locked motor. The position thus remains constant. Only the locked motor is currently simulated, but other failure modes could result in runaway motors or free-spinning motors.

The robot's position, velocity, and acceleration calculated in this procedure are sent to the sensor routines. The robot position is also sent to the graphics simulator which displays the motion on the screen. This is the same graphics program used by the Trick simulation package.

3.4 Fault Detection Capabilities

3.4.1 Failure Modes

If a sensor breaks and the failure goes undetected, the host computer will be performing its calculations using erroneous information. In this simulator, the encoders break in a frozen mode continuously reporting the last value read before the failure. The tachometers fail by continuously reporting zero velocities and thus constant positions. With these failure modes, the host sees the error between the sensed angle and the desired angle grow for the joint with the faulty sensor, and the control equations increase the appropriate output torque to the robot to try and compensate for the error. The joint with the faulty sensor swings wildly off course because the host keeps trying harder and harder to get the broken sensor value to match the desired value. Since the calculations for all the joints are based on knowledge of where the other joints are located, all of the other output torques are also computed incorrectly and the joints all stray from their desired paths.

When a motor fails, it locks in position and the joint is then unable to move. If a motor failure goes undetected, the sensors are still reading the correct information. In reality, the motor failure would probably result in a sensor failure as well, but the result would still be that both sensors are reporting a constant joint angle. The control equations try to push the broken joint closer to the desired value but the frozen motor does not respond to the torques. Since the sensors are still reporting the actual position of the joint, all the other calculations are based on correct data and the other joints can continue with their normal motions. The plan must be modified, however, to get the end effector to its desired location.

3.4.2 Thresholds

These two undetected failures reveal the importance of getting accurate sensor readings and of detecting a sensor failure quickly. A frozen motor is not as critical a failure in most cases and can be dealt with at a more leisurely pace. Since the sensors are not perfectly accurate, an acceptable threshold for the error between sensor reading and desired value must be chosen. Unfortunately, even during normal operation, the error between the actual angle and the desired angle can be relatively large especially at the beginning of a run before the controller has had time to bring the error under control. Choosing the maximum error found during a failure-free run results in a threshold that is so large, it may take several time steps to notice the error from a broken sensor. By the time the failed sensor is detected, the robot controller has already been infected with the erroneous information and the robot is either off course or has damaged itself.

Fortunately, the error between the angles recorded by the two sensors during normal operation is very small even after integrating the tachometer reading to get the angular position. Modeling errors and errors induced by unpredicted loads affect both sensors in a similar manner. Thus, a tight threshold can be chosen for a comparison of the two sensed positions. If this threshold is exceeded, the fault detection software assumes that one of the sensors has failed and appropriately chooses one as the working sensor from which to take the recorded data. The larger thresholds from the typical error between the sensed and

desired angles are still monitored, however. The large thresholds provide a means of checking for a motor failure.

The pseudo-code for these checks is reproduced below. The angle θ_d and its derivatives are the desired values. The variables θ_t, $\dot{\theta}_t$, and $\ddot{\theta}_t$ are the values derived from the tachometer reading. The results based on the encoder are θ_e, $\dot{\theta}_e$, and $\ddot{\theta}_e$. Finally, θ and its derivatives are the values sent to the robot controller.

```
If ((encoder working) and (tachometer working)){
    θ = θ_e, θ̇ = θ̇_t, θ̈ = θ̈_t
    If ((|θ_t − θ_e|) >= threshold){
        if (encoder working){
            tachometer = failed
            θ = θ_e, θ̇ = θ̇_e, θ̈ = θ̈_e
        }else{
            encoder = failed
            θ = θ_t, θ̇ = θ̇_t, θ̈ = θ̈_t
        }
    }else{
        if ((|θ_d − θ_t|) >= tachometer-threshold)
            tachometer = failed
        if ((|θ_d − θ_e|) >= encoder-threshold)
            encoder = failed
    }
}
If ((tachometer == failed) and (encoder != failed)){
    if ((|θ_d − θ_e|) < encoder-threshold){
        θ = θ_e, θ̇ = θ̇_e, θ̈ = θ̈_e
    }else{
        encoder = failed
        motor = failed
        send stop motor signal to robot
    }
}
If ((encoder == failed) and (tachometer != failed)){
    if ((|θ_d − θ_t|) < tachometer-threshold){
        θ = θ_t, θ̇ = θ̇_t, θ̈ = θ̈_t
    }else{
        tachometer = failed
        motor = failed
        send stop motor signal to robot
    }
}
```

Choosing which sensor has failed and which is still working when the tight tolerance is exceeded is the most difficult task. Intuitively, one would expect the sensor with a reading closer to the desired value to be the working sensor and would switch to obtaining all the information from that sensor. However, experience has shown that the fault detection software choses the correct sensor only when the desired values are increasing. If the desired angles are decreasing in value, it consistently picks the failed sensor as the working one.

This problem is a result of the time it takes the controller to bring the errors under control and the failure modes for the sensors. Both the encoders and the tachometers fail by reporting a constant angular position either directly or by integration of a zero velocity. First, let us assume the sensors always read less than the desired value. If a sensor fails and gets stuck at a specific value while the desired values are increasing, the error will grow and the fault detection routine should take the angle information from the sensor that reads closer to the desired value. However, if the sensor fails while the desired values are decreasing, the desired values are approaching the failed value. The error starts decreasing and the surviving sensor is often the one whose absolute error is larger. The opposite relationships hold if both sensors are reading values greater than the desired angle. A failed sensor would then have the smaller error during an increase in desired angles and the larger error during a decrease in desired angles.

The various sensor failure situations that arise in the presence of increasing desired values are listed in Table II. The variable d_t is the tachometer error or the absolute difference between the desired value and the tachometer value. Similarly, d_e is the encoder error. The bold faced entries are the actual sensor failures which occur given the specified orderings of the angles and sensed angle errors. The entries enclosed in parentheses are the action taken by our current fault detection algorithm which takes into account the ordering situations described in the preceding paragraph. The intuitive, more naive algorithm would always choose the encoder as the survivor for the case where $d_t > d_e$ and would always choose the tachometer otherwise. The table for decreasing desired values would look similar to Table II but would shut down the joint in the opposite column.

Table II: Failure Situations and Detection Actions for Increasing Desired Angles

Angle Ordering	Error Ordering	
	$d_t > d_e$	$d_e > d_t$
$\theta_d < \theta_t, \theta_e$	**Encoder Failed** (choose tach)	**Tach Failed** (choose encoder)
$\theta_t, \theta_e < \theta_d$	**Tach Failed** (choose encoder)	**Encoder Failed** (choose tach)
$\theta_t < \theta_d < \theta_e$	**Encoder or Tach** (Shut Down Joint)	**Encoder Failed** (choose tach)
$\theta_e < \theta_d < \theta_t$	**Tach Failed** (choose Encoder)	**Encoder or Tach** (Shut Down Joint)

By checking whether the desired values are increasing or decreasing and performing the appropriate comparisons to choose the surviving sensor, the fault detection algorithm can correctly isolate the failed sensor in 75% of the cases instead of 50% for the naive algorithm. The remaining 25% of the cases are inconclusive as either sensor failure could produce the same sensed angle ordering for the given order of the angles. In the case where $\theta_e < \theta_d < \theta_t$ and θ_d is increasing, for example, an encoder failure would result in the encoder error growing larger while the tachometer error still tracks the desired value. Thus, d_e would most likely be greater than d_t. However, if the tachometer failed, the desired value approaches the static tachometer value, and d_e would again be greater than d_t (assuming the angle ordering does not change). Both failures result in the same ordering of the sensor errors. The algorithm shuts down the appropriate joint to avoid choosing the wrong sensor which would feed erroneous information to the controller and cause other joints to swing off course.

The fault detection algorithm thus provides the robot with fault tolerance of most single sensor failures by obtaining the angle information solely from the surviving sensor. Through the detection and isolation of a sensor failure, the algorithm is able to make use of the redundant information provided by the other sensor. When the algorithm cannot isolate the failure, it still protects the system from the hazards of faulty sensor readings.

In general, our simple fault detection simulator is capable of detecting for each joint a single sensor failure, a single sensor failure followed by a motor failure, or a motor failure. The simulator will eventually detect a second sensor failure and will catch the single failures it has missed, but it has allowed enough erroneous sensor readings through to the controller that other joints have been knocked off course and fail as well. In order to improve the fault detection algorithm, we must switch from the hard-coded voting scheme presented above to other forms of analytical redundancy which use filters [12,17], adaptive thresholds [8], or parity relations [5]. Willsky gives a thorough review of the various methods of analytical redundancy in [17] and [5]. Unfortunately, the amount of uncertainty and modeling errors present in most robotic control systems makes several of the proposed methods impractical. The generation of residuals using parity relations is one example of a method which would be unsuitable for robotic applications [8]. Most of the work in analytical redundancy has been focused on failure detection in aircraft, power generation system and other mechanical systems [13]. The algorithms for these systems will need to be modified for robotics applications.

4 Conclusions and Future Work

In this paper we have presented new results in fault tree analysis and fault detection for robot manipulators. This research sets the stage for significantly enhanced activity in fault tolerance for robotics. Once a failure can be detected and isolated, a fault tolerant expert system like Robo-MEDIC can proceed with the appropriate actions to make use of the existing robot structure, redundancy, and alternate paths. There already exist a variety of computer fault tolerance schemes which can provide a starting point for creating these structurally independent robotic fault tolerance algorithms.

The fault tree analysis for the Riceobot has proven useful in pointing out some trouble spots for fault detection and fault tolerance. Even without a quantitative analysis, the importance of certain components and the severity of different failures are revealed in the fault trees. For robots, the good health of the internal sensors is shown to be extremely desirable. Erroneous data from even one sensor at a joint can cause the whole robot to deviate drastically from its course if the failure is not detected quickly. Without the sensors, the robot also loses much of its capability to detect faults. Developing methods for early detection of sensor malfunctions thus has a high priority in this research.

By simulating relatively simple fault detection situations, we are gaining a better understanding of how to satisfy this need for early detection. Our simulator has shown that to avoid false alarms due to modeling errors and noise we are forced to implement large thresholds which let some failures go undetected for too long. Other relationships must be developed concerning the information available in order to improve the fault detection algorithms. We will embed the algorithms in our expert system and integrate the simulation into the Trick simulation package to create a more flexible fault detection and fault tolerance simulator.

The analysis of the fault trees in this paper will be useful in creating fault tolerant algorithms. Through an analysis of the structures, fault tolerance schemes can be developed which will attempt to maintain the health of the internal nodes in the presence of failures in their children. These algorithms will rely only on the available components or structure of the robot and can be developed from the knowledge amassed during the fault detection simulations. The fault tolerant algorithms will also be tested on the enhanced Trick robotic simulation package.

Acknowledgments

This work was supported in part by the National Science Foundation under grants MIP-8909498 and MSS-9024391 and in part by a Mitre Corporation Graduate Fellowship and an NSF Graduate Fellowship.

References

1. Bailey, R. W. and Quiocho, L. J., TRICK SIMULATION ENVIRONMENT DEVELOPER'S GUIDE, NASA JSC Automation and Robotics Division, Beta-release edition, February 1991.
2. Bloch, H. P. and Geitner, F. K., AN INTRODUCTION TO MACHINERY RELIABILITY ASSESSMENT, Van Nostrand Reinhold, 1990.
3. Butler, R. W. and Stevenson, P. H., "The PAWS and STEM Reliability Analysis Programs," NASA TECHNICAL MEMORANDUM 100572, NASA Langley Research Center, March 1988.
4. Chean, M. and Fortes, J. A., "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays," IEEE COMPUTER, 23(1):55-67, January 1990.
5. Chow, E. Y. and Willsky, A. S., "Analytical Redundancy and the Design of Robust Failure Detection Systems," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, AC-29(7):603-614, July 1984.
6. Giarratano, J. and Riley, G., EXPERT SYSTEMS: PRINCIPLES AND PROGRAMMING, PWS-Kent Publishing Company, Boston, MA, 1989.
7. Hamilton, D., "Elec 590 Project Report," Advisors: I.D. Walker and J.K. Bennett, Rice University, Department of Electrical and Computer Engineering, Houston, Texas, April 1991.
8. Horak, D. T., "Failure Detection in Dynamic Systems with Modeling Errors," JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS, 11(6):508-516, November-December 1988.
9. Kota, K. and Cavallaro, J. R., "Run-Time Fault Detection and Isolation with the IEEE Std 1149.1 Test Access Port," In IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, San Diego, CA, May 1992. To be Submitted.

10. Maciejewski, A. A., "Fault Tolerant Properties of Kinematically Redundant Manipulators," In PROCEEDINGS 1990 IEEE CONFERENCE ON ROBOTICS AND AUTOMATION, pages 638–642, Cincinnati, OII, May 1990.
11. Martensen, A. L. and Butler, R. W., "The Fault-Tree Compiler," NASA TECHNICAL MEMORANDUM 89098, NASA Langley Research Center, January 1987.
12. Merrill, W. C., DeLaat, J. C., and Bruton, W. M., "Advanced Detection, Isolation, and Accommodation of Sensor Failures - Real-Time Evaluation," JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS, 11(6):517–526, November-December 1988.
13. Stengel, R. F., "Intelligent Failure-Tolerant Control," IEEE CONTROL SYSTEMS, 11(4):14–23, June 1991.
14. Stiffler, J. J. and Bryant, L. A., "CARE III Phase II Report - Mathematical Description," NASA CONTRACTOR REPORT 3566, NASA Langley Research Center, 1982.
15. Tesar, D., Sreevijayan, D., and Price, C., "Four-Level Fault Tolerance in Manipulator Design for Space Operations," In PROC. OF THE FIRST INTERNATIONAL SYMPOSIUM ON MEASUREMENT AND CONTROL IN ROBOTICS, Houston, TX, June 1990.
16. Walker, I. D. and Cavallaro, J. R., "Fault Tolerant Robotic Architectures and Algorithms," In SIAM INTERNATIONAL CONFERENCE ON INDUSTRIAL AND APPLIED MATHEMATICS, Washington, DC, July 1991.
17. Willsky, A. S., "A Survey of Design Methods for Failure Detection in Dynamic Systems," AUTOMATICA, 12:601–611, 1976.
18. Winokur, P. S., "Radiation-Hardened Circuits for Robotics," PROC. FOURTH TOPICAL MEETING ON ROBOTICS AND REMOTE SYSTEMS, pages 311–315, February 1991.
19. Wu, E., Diftler, M., Hwang, J., and Chladek, J., "A Fault Tolerant Joint Drive Systems for the Space Shuttle Remote Manipulator System," In IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, Sacremento, CA, April 1991.

A NEURO-COLLISION AVOIDANCE STRATEGY FOR ROBOT MANIPULATORS.

Joel P. Onema and Robert A. Maclaunchlan*
Department of Electrical Egineering and Computer Science
*Department of Mechanical Engineering and Industrial Engineering
Texas A & I University, Kingsville, Texas 78363

ABSTRACT

The area of collision avoidance and path planning in robotics has received much attention in the research community. Our study centers on a combination of an artificial neural network paradigm with a motion planning strategy that insures safe motion of the Articulated Two-Link Arm with Scissor Hand System relative to an object. Whenever an obstacle is encountered, the arm attempts to slide along the obstacle surface, thereby avoiding collision by means of the local tangent strategy and its artificial neural network implementation. This combination compensates the inverse kinematics of a robot manipulator. Simulation results indicate that a neuro-collision avoidance strategy can be achieved by means of a learning local tangent method.

I. INTRODUCTION

The problem of collision avoidance in robotics[8] Bradley, Hollerbach, Johnson and Lozano Perez can be described as follows: given a starting and a goal configuration of some object or linked group of objects in an environment cluttered with obstacles, find a path of connecting line segments from the starting to the goal configuration, such that the object to be moved follows this path without interfering with any obstacle of the environment.

Traditionally [1], designing a robot control system involves two steps first a set of kinematic equations which express the physical constraints of the robot are derived, and second, a computer program model is implemented generating arm configuration sequences that move the robot's end-effector from its current position to a target position. While simulation runs works well in a laboratory, they often suffer from serious limitations when applied in realistic environments. Wear and tear on mechanical parts changes the kinematics of the robot manipulators and sensor characteristics tend to wander with time. When such changes occur, the control program must be updated or the robot must be maintained. The response time slows down when the degree of difficulty increases by unpredictable obstacles with different shapes or change of position in the workspace. A high degree of autonomous adaptive learning is the ultimate approach to consider when solving these major impairments. This paper focuses on solving the inverse kinematics of an articulated two-link arm with a scissor hand system which can beconsidered as embody by a sensitive skin type sensors [1]. We study the following problem: given the position and orientation of the manipulator, calculate all possible sets of joint angles over time, which could be used to attain a given object position and orientation. The initial solution to the problem enables the manipulator to attain its goal without a planned strategy (no collision avoidance). The final strategy used combines motion planning and artificial neural networks. This strategy is to insure a collision-free motion of the robot manipulator. Simulation results are presented and a possible extension to this work is discussed. This paper is an extension to the work reported in [1, 4].

II. AN ADAPTIVE LEARNING APPROACH

The subject of neural network is hardly new[14], but there has been much more recent[11] progress in developing methods for training more complex configurations of these networks. The simplest net is the perceptron [13, 14], whose training procedure can be called an error-correcting procedure. The weights are revised whenever a mistake is made. Both the output and the correct answers are expressed as 0 or 1 else an error occurs. For the perceptron, once the weighted sum is computed, the activation of the output unit is determine by a threshold logic (0 or 1) depending whether the threshold value was exceeded. The thresholds activation function creates nonlinearity. When The Classes of behavior, etc. are linearly separable the perceptron will find a line or a plane that yields no error. It concentrates on reducing errors, but may perform poorly when classes of behavior etc, are not linearly separable.

In the LMS[12] (Least Mean Square) learning system, the correct answers are still express in terms of 0 and 1, however they are not restricted to a binary values but can have continuous real values. The goal of the LMS training procedure is to minimize the average (squared) distance from the true answer to the output. The LMS training procedure performs relatively well when classes are not linearly separable, because the best line of separation is found in terms of the minimum error

distance. To compute error rates, decision criteria are needed to determine the class that is selected. For a single output this is the class that is closest to the output. The error distance to the correct answer can be small, while erroneous answers may have larger distance or be on the wrong side of the boundary. Eventually, LMS training should converge to the minimum distance. This iterative technique of minimization is also called gradient descent[12]. The weights are constantly adjusted in the direction of the greatest reduction of error, example we expect that we are moving "down hill" in the direction of the minimum. The main problems for gradient descent methods is oscillation and not converging or poor convergence rate. The rate of convergence highly depends on the learning rate[11, 12]. The LMS learning system is completely linear. The weighted sum is directly used to activate the output unit; no additional mapping by the activation function is made. A linear function would not be useful, because the overall system would still be linear and would not effectively use the hidden units. While the threshold logic of the activation function perceptron is nonlinear, from the mathematical view point, the LMS needs a continuously differentiable function. This is not the case for threshold logic. Thus an alternative is activation function which is differentiable is used known as a logistic or sigmoidal function. For any real valued numbers, the activation function is a continuous value, this is a valid range in probability, these functions have been widely used in statistics.

Both the perceptron and the LMS learning System attempt to derive a linear separator from labeled data. Perceptron and LMS can be described as single layer neural networks, where a layer represents a set of output devices. The weights are termed feed forward, because they flow in the forward direction. Starting with the inputs node and weights, no weight cycles back to an input node or output node of previous layers.

Very few real-world applications are truly linearly separable this basic discussion is beyond the scope this paper. The multilayer neural network can be trained as by a generalized version of the LMS training procedure for a nonlinear logistic outputs known as Backpropagation [11]. We consider this paradigm to be the most suited for our needs. It is flexible and can be easily implemented. It is one the most popular and widely used neural networks today. Backpropagation has the ability to learn mappings by example, by a process of learning [1, 3, 5, 9, 10]. It uses a learning procedure that aims to minimize the mean squared error between the actual outputs of the network and some desired outputs. Because of convergence problems that found in the LMS and the learning rate constraints as previously mentioned we opted for modular[1] approach. It is much faster to train several smaller networks than one large one. We kept the number of neuron as small as possible, since removal of redundant neurons can make noticeable difference in the training time. The learning strategy that permits to solve the inverse kinematics problem [5, 7] and avoid obstacles is based upon a simple theoretical robot kinematics[1, 3] as presented in (fig. 1) and the value of the local tangent as presented in[2].

III. LOCAL TANGENT

The robot arm attempts to slide along the obstacle surface as reported in [2] (Fig. 2), whenever an obstacle is encountered. This sliding is accomplished by a coordinated move between joints J1 and J2, based on the value of the local tangent (Fig. 4). The local tangent is obtained from the following formulas depending on were the obstacle occurs in the work space. There are three categories (Fig. 3). Note: a complete derivation of the following equation can be found in [2]

Type I are those obstacles that obstruct link 1. Since link 2 is irrelevant in this case, then $d\theta_1=0$ and $d\theta_2 \neq 0$. The local tangent for

Type I is vertical [2].

Type II consists of the obstacles that obstruct link 2. Assume that link 2 is obstructed at point C by a type II obstacle, at the distance L_d from the joint J2 (Fig. 4). Then, the estimates [2] of $d\theta_1$, and $d\theta_2$ at C can be found as follows:

$$c_x=L_1\mathrm{COS}(\theta)+L_d\mathrm{COS}(\theta_1+\theta_2)$$

x coordinate of C (1)

$$c_y=L_1\mathrm{SIN}(\theta_1)+L_d\mathrm{SIN}(\theta_1+\theta_2)$$

y coordinate of C

The expression for the local tangent of type II at point C is given by the estimate [2]:

$$\frac{d\theta_2}{d\theta_1}=\left(\frac{L_1}{L_d}\mathrm{COS}\theta_2+1\right) \qquad (2)$$

Type III are obstacles that obstruct the wrist. Sliding of the wrist P along the obstacle corresponds to its moving along the line segment LM (Fig. 4). β_1 is the angle between the line perpendicular to link 2 and the line from P to the obstacle and β_2 is the angle between the line LM and the positive x axis. Then $\beta_2 = \theta_1+ \theta_2+ \beta_1-\pi$. The expression for the local tangent estimate [2] of type III appears as:

$$\frac{d\theta_2}{d\theta_1}=-\left(\frac{\frac{L_1}{L_2}+\mathrm{COS}\theta_2+\mathrm{SIN}\theta_2\tan(\theta_2+\beta_1)}{\mathrm{COS}\,\theta_2+\mathrm{SIN}\theta_2\tan(\theta_2+\beta_1)}\right) \qquad (3)$$

IV. LEARNING STRATEGY

The Neural Network is trained to map the Cartesian coordinate to the joint angle coordinate transformation for the three degree of freedom robot arm. The net learns the topology and unknown transformation from presentation of examples such that a solution to the inverse kinematics is found and an obstacle can be avoided. First, the network is presented with a set of examples for training, then tested to generate an output within an acceptable tolerance. Our study centers on the following mapping: given the Cartesian position (x, y),

generate the accurate joint angles. Secondly, the net is trained and tested to recognize the obstacle types, (I, II, III). This permits the usage of the local tangent and the generation of a collision avoidance motion, in other words, the arm learns to slide along the obstacle surface with no contact at the tangential point. A combination of the modules described above can be an important tool that can carry out the extensive computation and planning needed to achieve an intelligent move, therefore avoiding an obstacle.

A possible real-time architecture [2] implementation of this study is presented in Fig. 10. Sensitive Skin sensor information and arm position are passed to the neural planner/ controller that generates a safe motion of the arm.

V. SIMULATION RESULTS

As already stated, we opted for a modular approach, since it is much faster to train several smaller networks than one large net. Fig. 5 shows the general network configuration followed by subnets configurations. The subnets configurations are three layer networks ranging from two to five nodes at the input layer and six to twelve nodes for the middle layer and raging from one to three nodes for the output layer. We have trained and tested the nets for a complete mapping of the kinematic transformation of the experimental arm/hand system 3 degree of freedom (dof). In our experiments [1] we focused on training the net by mapping the joint angles (shoulder, elbow and wrist) and their joint position. Our experiment has indicated an ease in training the wrist, the elbow and shoulder, because of the modularity and proper data normalization. Our first experiment focused on the kinematics transformation when there is no obstacle. Maximum and RMS (Root Mean Square) error values for a case study of 2 dof (fig. 6a, 6b) illustrate a downward trend, indicating the smoothness of the learning curves. Fig. 7a shows the expected joint position results graph from the examples presented to the network by training. Fig. 7b shows a graph depicting the network output when tested. As one might observe the network performance is within tolerable range when compared to the expeted results graph. The second experiment targets the type II obstacle using simulated sensor data. Fig. 8a shows the expected results graph from examples presented to the network via training and fig. 8b depicts the network output when tested. The network output is an adaptation of the joint angles as the robot manipulator moves from an initial to final configuration space by sliding along the obstacle surface. As one might observe the network performance is also within tolerable range when compared to the expected results graph.

VI. CONCLUSION AND FUTURE PROJECTION

In this paper we have shown that a neural network paradigm can compensate the inverse kinematic behavior of our model and promising results of the neuro-collision avoidance strategy have been obtained. Our experiments have indicated that a modular approach is the most appropriate. The network was able to learn and mimic a decision making strategy by means of the learning tangent approach, thereby allowing the manipulator to achieve a collision free motion. Real-time implementation of this experiment will require a parallel architecture that can be of great benefit to the enhancement of the overall system performance. Because of the sensor data complexity, in future work the Learning Tangent method could be compensated by a fuzzy logic system [15] that bases its decisions on inputs in the form of linguistic variables, for example smooth, slippery and rough. In our case it will be the obstacle types (I, II, III). Our ultimate goal is to enhance the capabilities of the experimental arm/hand system by means of applicable findings leading to its real-time implementation. Current applications could include the NASA EVA Retriever and related space operation. In manufacturing, the learning tangent can play an important (safety) role in discerning the sudden presence of an operator in the working space resulting from careless behavior.

REFERENCES

[1] Joel P. Onema, A Motion Planning Strategy and Neural its Network Implementation for an Articulated Two-Link Arm with Scissor Hand System, MS Thesis, EE&CS Department Texas A&I University, May 1991.

[2] Cheung, E and Lumelsky, V. '' Proximity sensing in Robot Manipulator Motion Planning System and Implementation'', IEEE Conference on Robotics and Automation, Vol. 5 No. 6. December 1989. pp. 740-751.

[3] Josin, G. , Charney, D. and White,'' A Neural Representation of an Unknown Inverse Kinematics, Transformation'', Neural Systems Inc. 1989, Vancouver, B. C. Canada.

[4] Joel P. Onema, Barbara S, Schreur, Robert A. Maclaunchlan, Muhammed Ashtijou, '' A Motion Planning Strategy and its Neural Network Implementation for an Articulated Two-Link Arm with Scissor Hand System", IJCNN-91. Seattle, Washington.

[5] Guez, A. and Ahmed, Z. ''Solution to the Inverse Kinematics Problem in Robotics by Neural Networks'', Proceeding of the Second InternationalConference on Neural Networks, San Diego, California, 1988, Vol. 2, pp. 14-20.

[6] Richard P. Paul,'' Robot Manipulators:Mathematics, Programming, and Control'', Cambridge , MA: MIT Press 1986.

[7] Guo, J. and Cherkassky, V. ''A Solution to the Inverse Kinematics Problem in Robotics Using Neural Network Processing. Proceeding of the IJNCNN-89, Washington DC, Vol 2.PP. 289-304.

[8] Brady, M. J , Hollerbach, T., Johnson, T. Lozano- Perez ''Robot Motion Planning'', MIT Press, MA. 1982.

[9] Judith E. Dayhoff,''Neural Network Architectures'', Van Nostrand Reinhold, NY,1990.

[10] Kung, S. Y. and Hwang, J. N. "Neural Network Architecture for robotics". IEEE Transaction on Robotics and Automation. 1989. Vol. 5. Nr. 5. PP. 641-657.

[11] Rumerlhart, D.,E. and McClelland, J. L., Parallel Distributed Processing: Explorations in Microstructure of Cognition, vol. I , MIT Press, 1986.

[12] Widrow, B., and Stearns, S. D. Adaptive Signal Processing , Prentice-Hall, Inc., Engelwood Cliff, NJ, 1985.

[13] R.P. Lipman: '' An Introduction To Computing With Neural Nets'', IEE ASSP Magazine, April 1987, pp.4-22. perceptron.

[14] Minsky M. L. S. Paper, Perceptrons: ''An introduction to computational Geometry". MIT Press, Cambridge, Mass 1969.

[15] Bart Kosko ,'' Neural Network and Fuzzy Systesms '', Prentice hall, Inc. Engelwood Cliff, NJ, 1991.

KINEMATICS EQUATIONS

Forward transformation:
$X= L_1COS\theta_1+L_2COS(\theta_1+\theta_2)$ and

$Y= L_1SIN\theta_1+L_2SIN(\theta_1+\theta_2)$

Inverse transformation:

$COS\theta_2=(X^2+Y^2-L^2_1-L^2_2)/2L_1L_2$

$\theta_1=\arctan(X/Y)\arctan[L_2SIN\theta_2/(L_1+L_2COS\theta_2)]$

L_1 : length of first link

L_2 : length of second link

θ_1 : joint of first link

θ_2 : joint of second link

Fig. 1

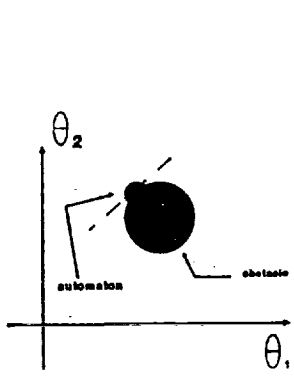


Fig. 2

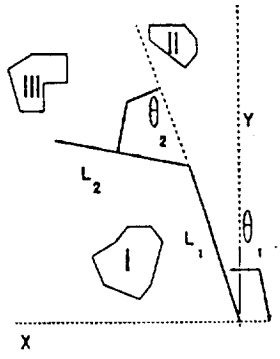


Fig. 3

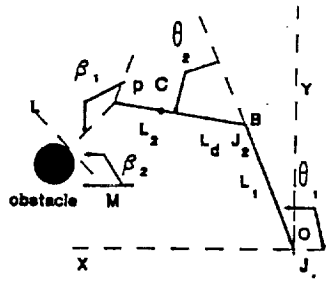
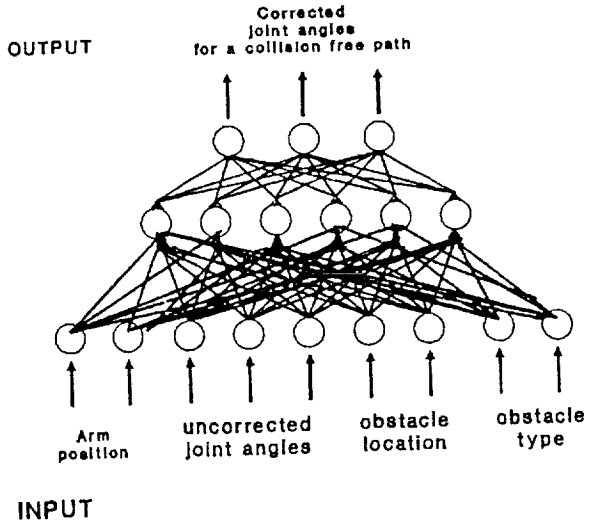


Fig. 4



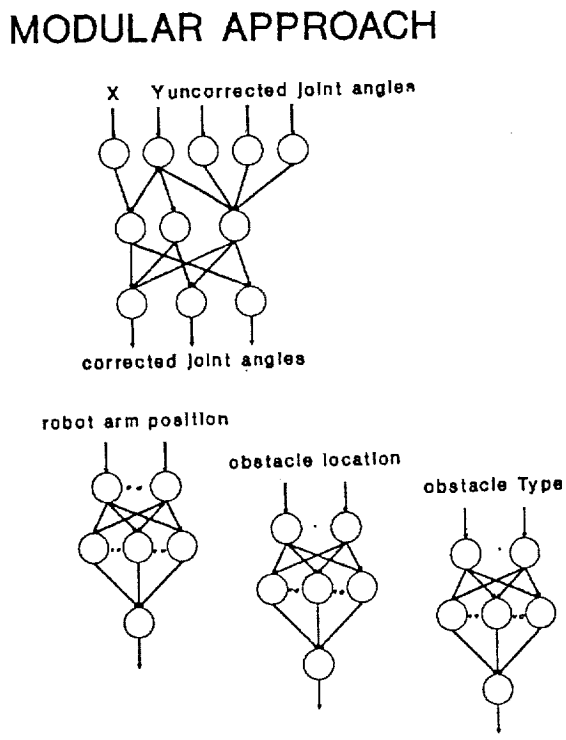


Fig. 5.

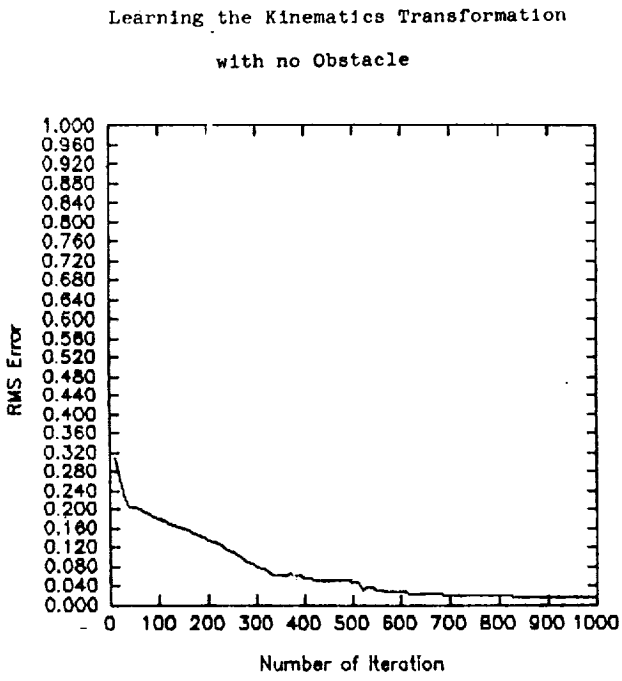


fig. 6a.

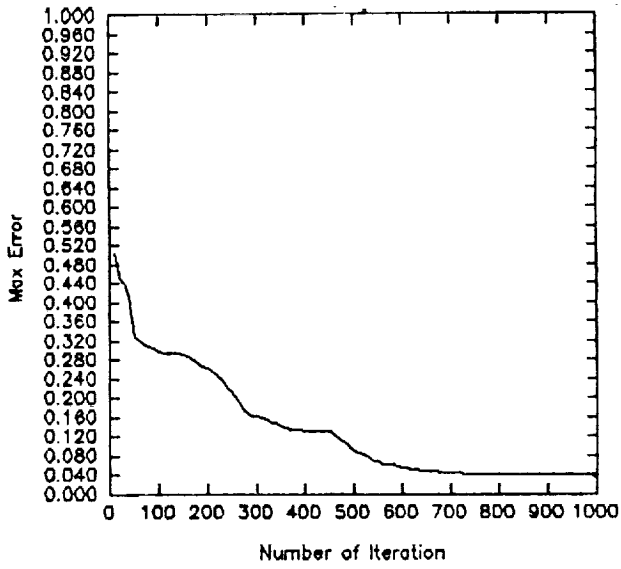
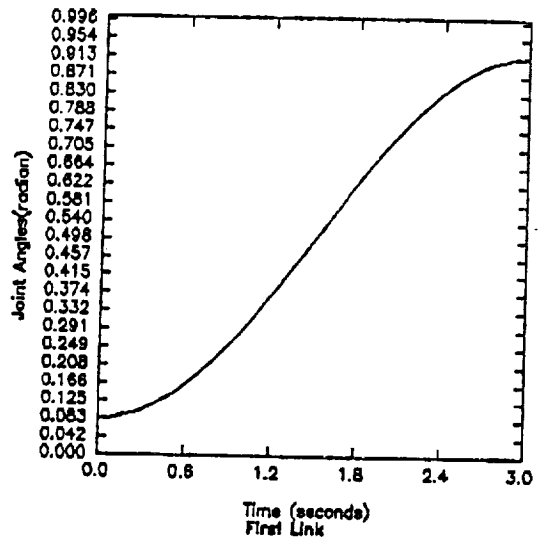


fig. 6b.

Expected Output



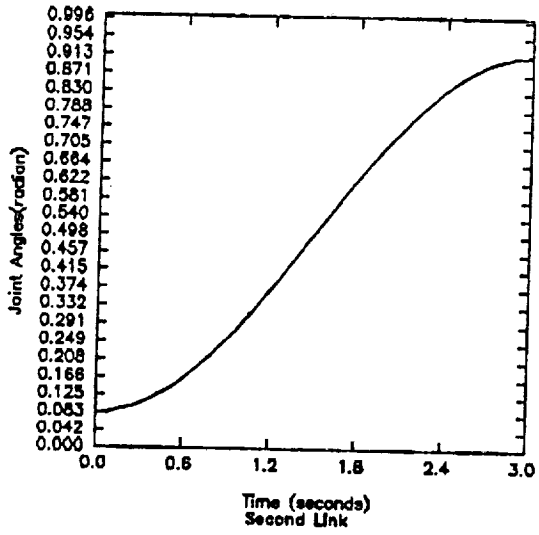
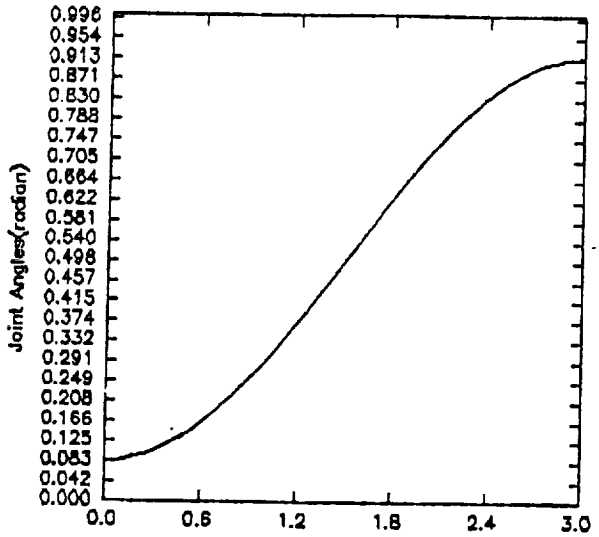


Fig. 7a.

Network Output



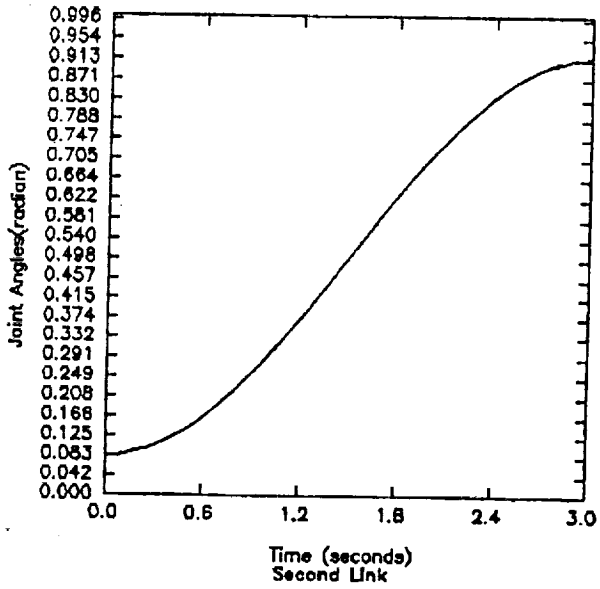


Fig. 7 b

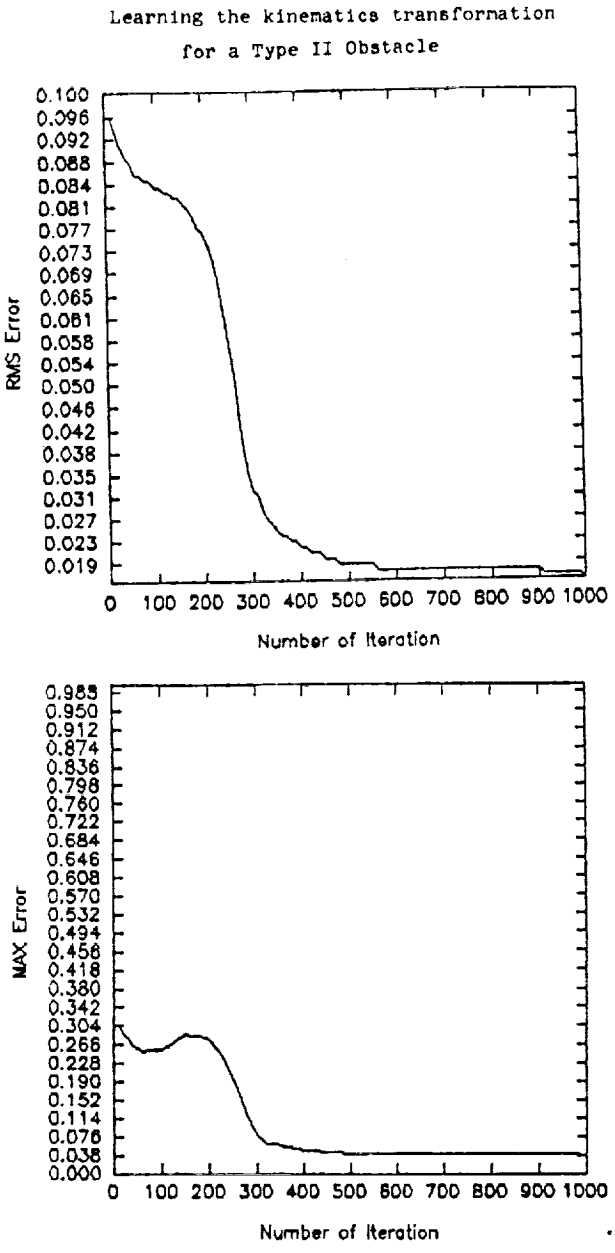

Learning the kinematics transformation
for a Type II Obstacle
RMS Error
0.100
0.096
0.092
0.088
0.084
0.081
0.077
0.073
0.069
0.065
0.061
0.058
0.054
0.050
0.046
0.042
0.038
0.035
0.031
0.027
0.023
0.019
0 100 200 300 400 500 600 700 800 900 1000
Number of Iteration
MAX Error
0.988
0.950
0.912
0.874
0.836
0.798
0.760
0.722
0.684
0.646
0.608
0.570
0.532
0.494
0.456
0.418
0.380
0.342
0.304
0.266
0.228
0.190
0.152
0.114
0.076
0.038
0.000
0 100 200 300 400 500 600 700 800 900 1000
Number of Iteration

Fig. 8.

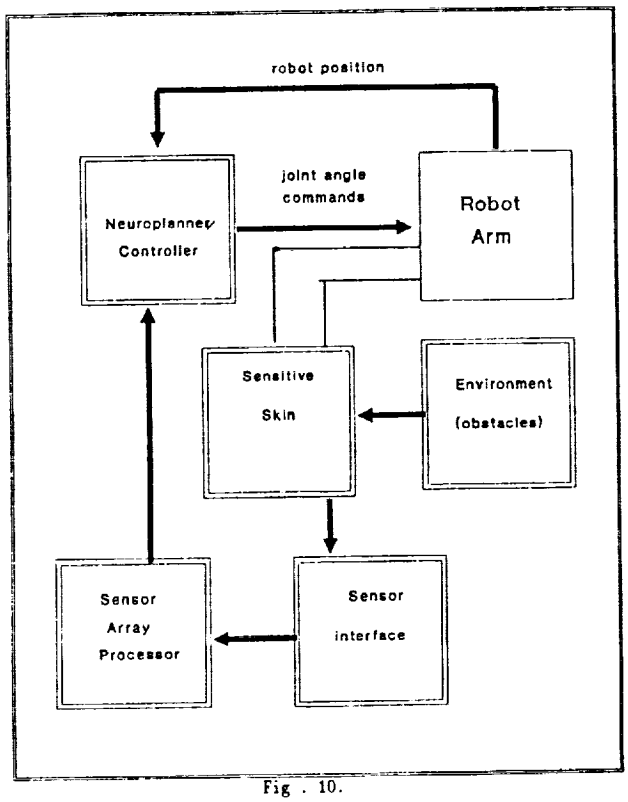
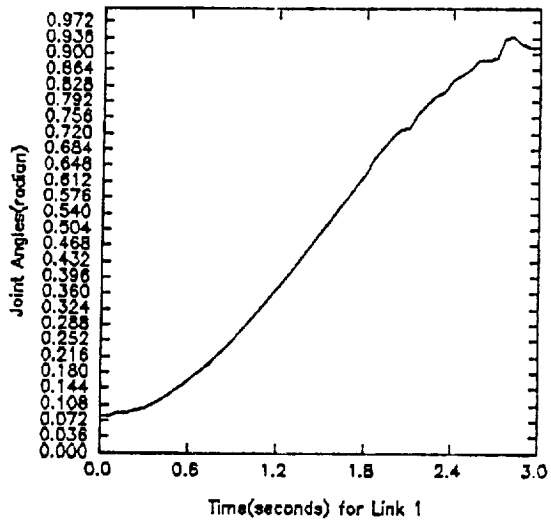

robot position
Neuroplanner/
Controller
joint angle
commands
Robot
Arm
Sensitive
Skin
Environment
(obstacles)
Sensor
Array
Processor
Sensor
interface

Fig . 10.

expected output



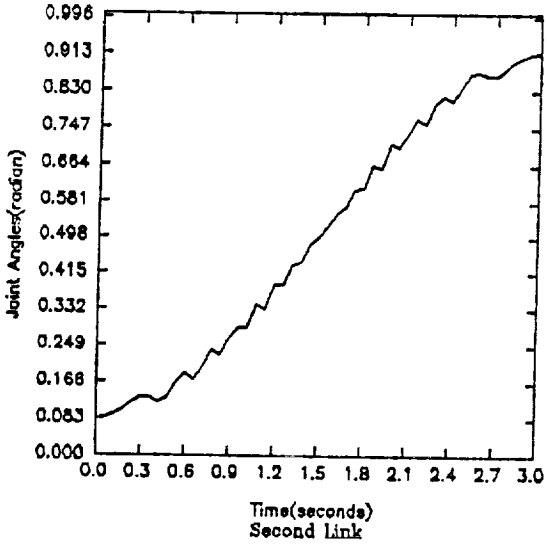
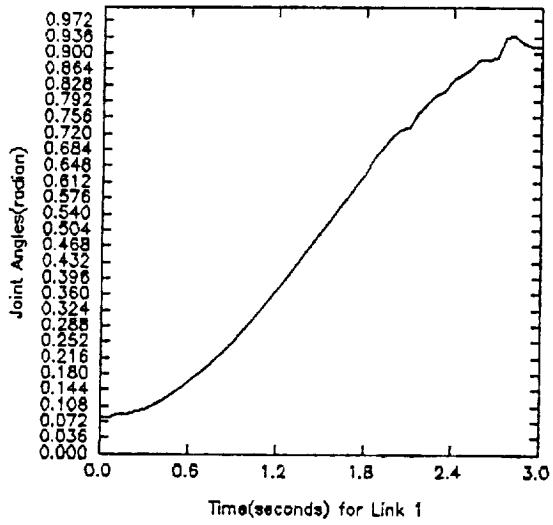


Fig. 8a.

Network Output



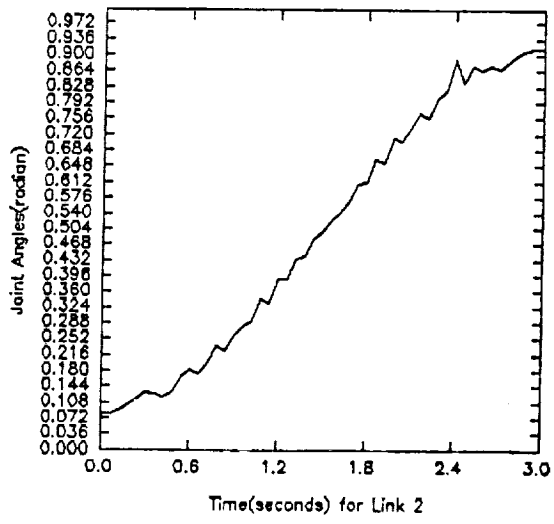


Fig. 8.b.

N93-11963

Interactive Collision Detection

Brad Bell
Barrios Technology Inc.
Houston, Texas

The assembly of Space Station Freedom will require operations where large bodies are manipulated in close proximity to one other. A fast and reliable method for performing collision detection would greatly benefit the development and verification of such operations. The Interactive Graphics and Operations Analysis Laboratory (IGOAL) has developed an algorithm for performing collision detection which provides accurate results at interactive speeds. This algorithm uses a highly-optimized ray tracer for performing the analysis. The purpose of this presentation is to describe the algorithm and demonstrate its capabilities.

Session A4: MANNED PERFORMANCE

Session Chair: Dr. Mark Stuart

AN 8-D.O.F. DUAL-ARM SYSTEM FOR ADVANCED TELEOPERATION PERFORMANCE EXPERIMENTS

Antal K. Bejczy and Zoltan F. Szakaly
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

This paper describes the electro-mechanical and control features of an 8-D.O.F. manipulator manufactured by AAI Corporation and installed at the Jet Propulsion Laboratory (JPL) in a dual-arm setting. The 8-D.O.F. arm incorporates a variety of features not found in other laboratory or industrial manipulators. Some of the unique features are: 8-D.O.F. revolute configuration with no lateral offsets at joint axes; 1 to 5 payload to weight ratio with 20 kg (44 lb) payload at a 1.75 m (68.5 inches) reach; joint position measurement with dual relative encoders and potentiometer; infinite roll of joint 8 with electrical and fiber optic slip rings; internal fiber optic link for "smart" end effectors; four-axis wrist; graphite epoxy links; high link and joint stiffness; use of an upgraded JPL Universal Motor Controller (UMC) capable of driving up to 16 joints. The 8-D.O.F. arm is equipped with a "smart" end effector which incorporates a 6-D.O.F. force-moment sensor at the end effector base and grasp force sensors at the base of the parallel jaws. The 8-D.O.F. arm is interfaced to a 6-D.O.F. force-reflecting hand controller. The same system is duplicated for and installed at the Langley Research Center.

INTRODUCTION

Most commercially available manipulators have been designed and built with a specific application and performance domain in mind. When it comes to application research and development of a more general nature which typically requires some extra motion dexterity combined with some extra reach and load handling capability together with high positioning accuracy and repeatability, most existing manipulators have shortcomings and have to be modified or rebuilt.

The purpose of the research and development work described in this paper is to build an end-to-end manipulator system to enable the performance of a broad range of realistic tasks not achievable by other manipulators. Of particular interest are tasks like the real-life simulation of the Solar Max Satellite Repair (SMSR) in teleoperation mode. As known, this satellite was not built for maintenance, and was still repaired in Earth orbit in the Space Shuttle bay by two EVA astronauts in 1984. The question now is: can the SMSR task be performed remotely by the use of an advanced telemanipulator system? If so, then what kind of performance can be expected for an SMSR-type work in teleoperation mode?

The SMSR type teleoperation work also raises a number of interesting and important issues regarding redundancy in the kinematics, sensing and control of manipulators and regarding the human operator interface to a redundant manipulator system.

The two 8-D.O.F. manipulators built by AAI Corporation for JPL in 1990 (and two more for LaRC in 1991) serve the purpose of enabling the experimental evaluation of application oriented performance issues briefly indicated above.

In the first part of the paper we summarize the mechanical features of the AAI 8-D.O.F.

arm. The control electronics, including some computational aspects, are briefly outlined in the second part of the paper.

AAI ARM MECHANISM

The Advanced Research Manipulator II (ARM II), Model 1520-8A, is an 8-D.O.F., redundant manipulator designed and built by AAI Corporation (Hunt Valley, MD) to support laboratory telerobotics R & D work. It has a 20 kg (44 lb) payload capacity at a full extension. A high payload to weight ratio is achieved through the use of lightweight graphite epoxy composite materials for the arm links, lightweight modular joints, and high torque servo motors. The ARM II is based on a modular joint design, which permits construction of a wide variety of revolute kinematic configurations. The modular joints are sized according to torque requirements and can be mated to links, regardless of twist angles.

The special kinematic feature of the 8-D.O.F. ARM II is the four-axis, gimballed wrist which allows singularity avoidance in a very wide configuration range and permits small angular changes of the end effector with little or no motion of the lower arm joints. Another special feature of the design is the infinite roll capability of the last, 8th joint to which the end effector is mounted. This permits contiuous rotation about the roll axis of a tool held by the end effector without requiring motion of the other joints.

The ARM II is driven by DC brash motors with integral brakes and encoders. Harmonic drives are used as gear reducers. Each joint is equipped with two encoders for input and output position sensing in the harmonic drives. The encoders are relative position encoders. In addition, each joint is equipped with a potentiometer for sensing absolute position at start up. Each joint also has a thermal sensor, electronic limit switch (except the 8th joint) and a mechanical stop (again, except the 8th joint) outside the limit switch. Some of the motor, brake and gear characteristics are listed in Table 1. Figure 1 shows ARM II, Model 1520-8A as installed at JPL for system integration. Figure 2 shows the cabling system of the arm.

High joint stiffness is achived by the use of high stiffness ball bearings and specially stiffened harmonic drives. Together the two bearings, which have been used in previous space applications, protect the joint against thrust and radial loading. Links 3 and 6 incorporate tubular graphite epoxy elements which provide high stiffness, strength, low weight and structural damping. This last point was experimentally verified on a General Electric P50 robot arm and described in Ref. 2.

The ARM II joint reference frames, following the Denavit-Hastenberg (D-H) convention, together with the D-H parameters are listed in Figure 3. As seen, there are no lateral offsets of joint axes (the a_i D-H parameters are zero for all links). This greatly facilitates the handling of forward and inverse kinematics and dynamic computations. The total reach capability of ARM II is 68.5 inches with the JPL Model C Smart Hand. (Without end effector the reach capability is 60 inches).

Another important feature of ARM II is that stiffness and conservative design constraints allow it to be oriented as desired with respect to gravity. Nonetheless, ARM II is a lightweight manipulator when considering its 1:5 payload to weight ratio. Note that the PUMA 560 payload to weight ratio is 1:13. More on the ARM II design characteristics can be found in Ref. 1.

The mass and inertial characteristics of ARM II links are listed in Table 2. Table 3 lists the ARM II joint natural frequencies. These frequencies are based on motor and harmonic drive inertias without link load inertias since the gear ratios for each joint are sufficiently large to essentially eliminate the load inertia at the motor. Table 3 includes two natural frequencies for two joint stiffnesses: one valid for the very low torque range, K_a , and the other, K_b , valid over the remainder. The lowest natural frequency of the full system is calculated 5.3 Hz and is based on the vibration of joints 2 and 4 under conditions of full load at full extension and assuming that K spring constant applies. The system natural frequency with no payload and full extension

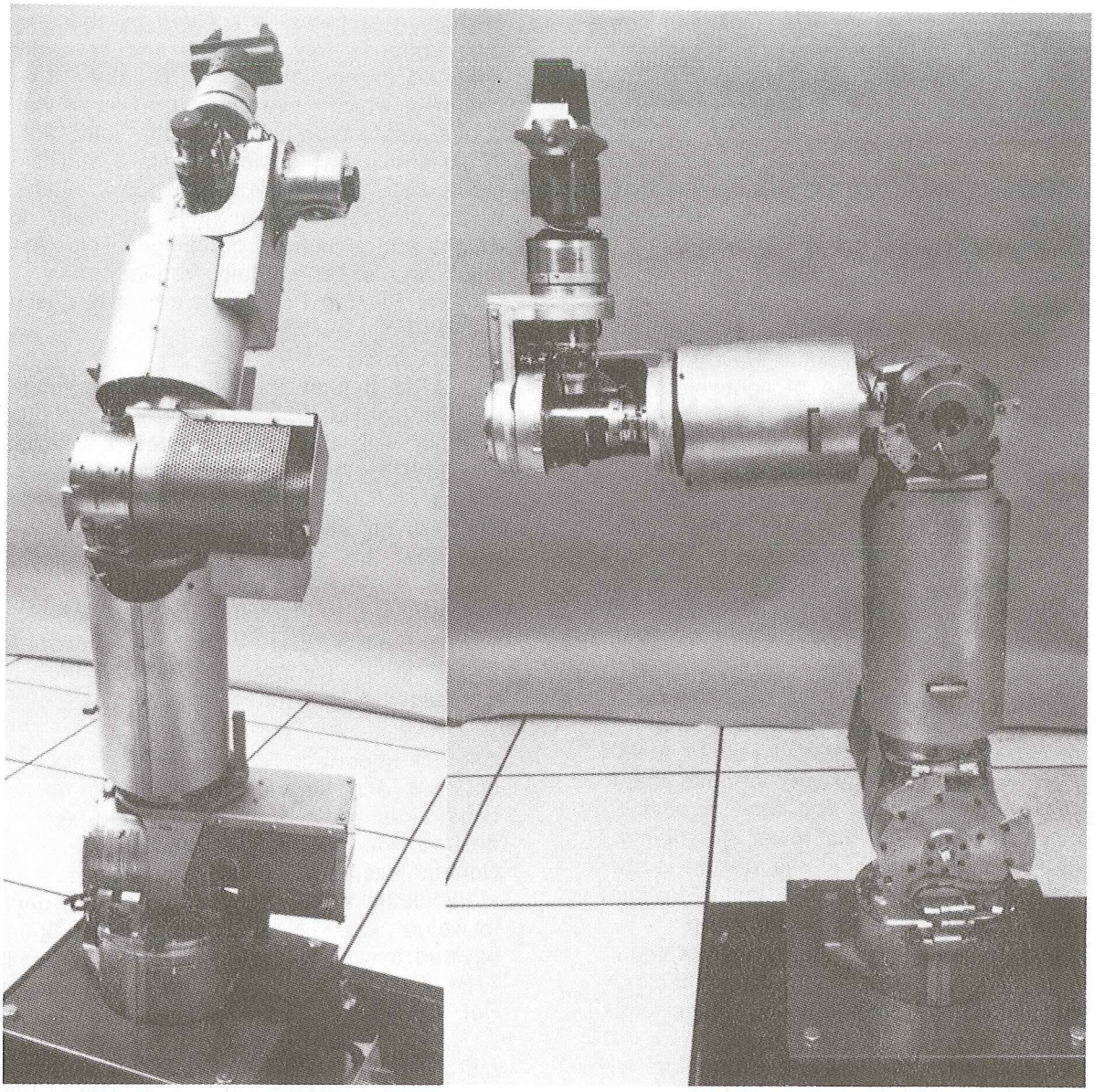


Figure 1. Eight D.O.F. ARM II (by AAI Corporation)

is calculated for 8.4 Hz. This compares well with the lowest natural frequency of the CM T3-776 robot arm measured by Tesar and Behi. This indicates the ARM II is similar to rigid, industrial robot arms in this regard.

The ARM II positional accuracy and repeatability under full load and at full extension is less than ± 0.1 inches and ± 0.01 inches, respectively. This is verified experimentally.

The measured maximum tip speed of ARM II at 30 VDC using only joint 1 motor at full extension was somewhat more than 18 inches per second.

In summary, ARM II incorporates a variety of features not found in other laboratory or industrial manipulators. Some of the unique features are:

- 8 D.O.F. revolute configuration
- Four-axis wrist
- No lateral offsets of joint axes
- High, 1:5 payload to weight ratio

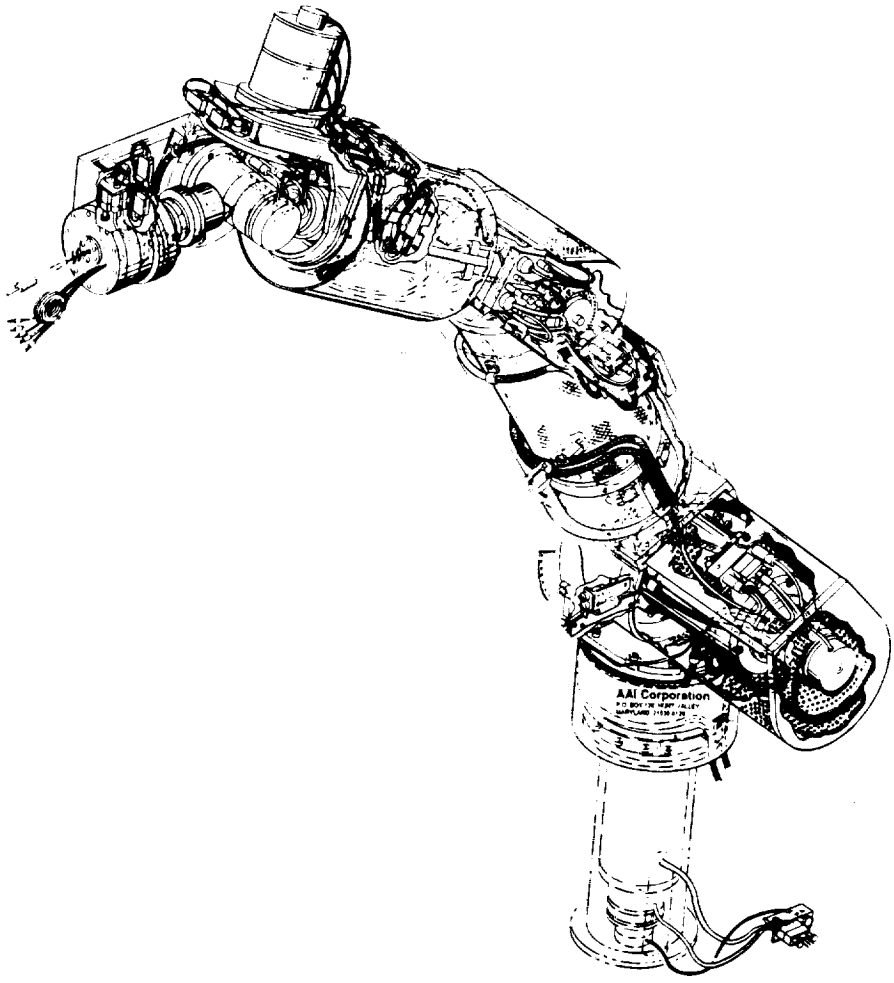
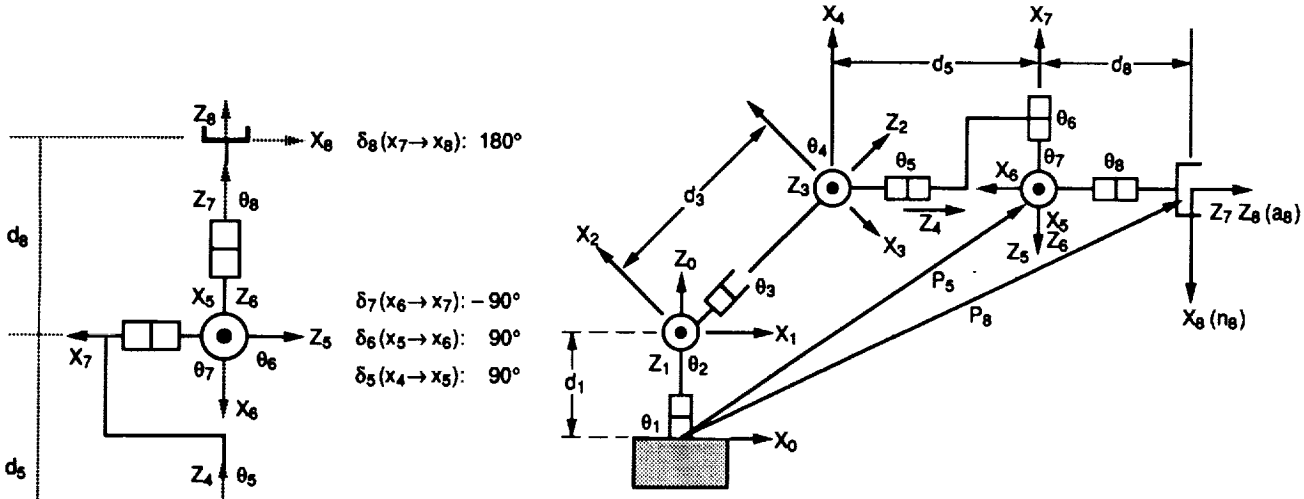


Figure 2. Eight D.O.F. ARM II Cabling Schematics

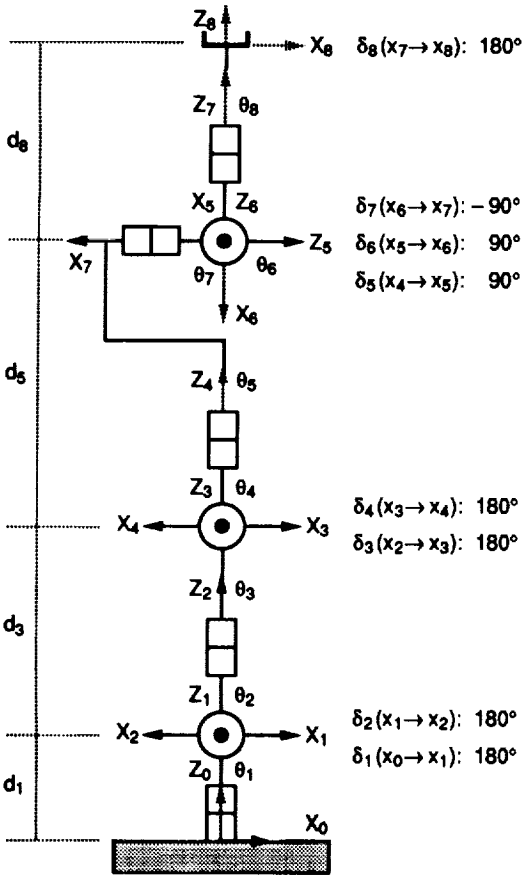
- 20 kg (44 lb) payload at maximum extension of arm (60 inches, without end effector)
- Modular joint design
- Graphite epoxy links
- High joint and link stiffness
- Infinite roll of joint 8 with electrical and fiber optic slip rings
- Internal fiber optic link for smart end effectors
- Two techniques for measuring joint position: dual relative encoders and potentiometer
- Indirect measurement of joint torque from dual relative encoders

CONTROL ELECTRONICS

The arms are controlled by a 16 axis UMC (Universal Motion Controller) each. The UMC was designed in our laboratory and it has been commercialized. The commercially available version is sold in four joint increments up to a 16 joint maximum per UMC. We use the commercial UMC ourselves for our various motor controller needs. The AAI arm has eight motors. Each joint, except the last one, is equipped with two optical encoders. There are a total of 15 encoders. These encoders count 4096 for every revolution of the motor shaft. The two encoders are connected to opposing ends of the harmonic drive. The gear reduction is 200 in every joint, so one encoder counts 0.5% slower than the other. Since both encoders are equipped with an index pulse, the two index pulses shift about 2 degrees relative to each other for every motor revolution. (Their relative position can eventually be used to determine absolute joint angle.) Since every joint also has a



THE POSITION VECTOR OF THE ORIGIN OF THE LINK 5 FRAME, P_5, IS UNIQUELY DETERMINED BY THE GIVEN POSITION VECTOR, P_8, AND THE APPROACHING VECTOR, a_8, OF THE END-EFFECTOR: $P_5 = P_8 - d_8 \cdot a_8$. NOTE THAT P_5 IS NOT A FUNCTION OF θ_5 .



DENAVIT-HARTENBERG (D-H) PARAMETERS

LINK i	JOINT VARIABLE θ_i	LINK DISTANCE d_i (mm)	LINK LENGTH a_i	TWIST ANGLE α_i	JOINT LIMIT
1	θ_1	d_1	0	90°	$-165° \le \theta_1 \le 165°$
2	θ_2	0	0	90°	$-105° \le \theta_2 \le 105°$
3	θ_3	696.1	0	90°	$-165° \le \theta_3 \le 165°$
4	θ_4	0	0	90°	$-105° \le \theta_4 \le 105°$
5	θ_5	557	0	−90°	$-165° \le \theta_5 \le 165°$
6	θ_6	0	0	90°	$-102° \le \theta_6 \le 131°$
7	θ_7	0	0	90°	$-130° \le \theta_7 \le 22°$
8	θ_8	474.1	0	0°	NO LIMIT

Figure 3. Zero Configuration of the Eight D.O.F. ARM II. The Joint Offset, δ_i , i = 1, ... , 8, are Defined as the D-H Angles at the Zero Configuration.

potentiometer, there are two ways to determine the absolute position of the arm. The two encoders provide two sources to determine the relative position. This is the primary quantity used for control. The load on the joint causes windup in the harmonic drive mechanism, this windup is precisely detected by the shift of one encoder position relative to the other. This is a way to determine joint torque. The alternative way is to read the motor current from the UMC.

The UMC is a cage housing two major subsystems, the multiprocessor and the motor control and sensing subsystems. (See Figure 4). Currently the multiprocessor subsystem consists of up to 8 processors. These are NS 32016 boards interconnected by a MULTIBUS -I backplane. These processors perform about 1 MIPS. All of our computations are currently done in this environment. We are developing a new high performance multiprocessor system that will be based on a custom designed high speed bus and processors in the 10 to 15 MIPS range each. We will describe that in more detail subsequently. The other major subsystem of the UMC is the motor controller. The motor controller consists of the following:

Joint processor
Joint interface
Power amplifiers
Input filters

The joint processor is one 32016 board dedicated to controlling the joints. It interfaces to the joint interface cards via a 16 bit i/o bus. This i/o bus is built to the

Table 1. Some Drive System Features of ARM II

Joint	MOTOR				Brake Rated Torque (oz-in)	Gear Ratio
	Rated Speed (rpm)	Rated Torque (oz-in)	Rated Current (A)	Rated Volt (V)		
1	1970	861	12.9	120	1200	200
2	1970	861	12.9	120	1200	200
3	2164	285	10.5	60	560	200
4	2164	285	10.5	60	560	200
5	1600	162	7.25	40	240	200
6	3000	86	7.8	35	128	200
7	1600	162	7.25	40	240	200
8	2050	51	4.9	30	240	200

Table 2. Mass/Inertia Parameters for Dynamic Model of AAI ARM II, Given in the Individual Link Reference Frames

Link & Reference Frame i	Mass (lb.s²/in) M_i	Mass Center (in) M_{X_i}	M_{Y_i}	M_{Z_i}	Moments of Inertia (in.lb.s²) I_{XX_i}	I_{YY_i}	I_{ZZ_i}	Reflected Rotor Inertia I_{a_i} ** (in.lb.s²)
0	X	X	X	X	X	X	X	356
1	.1612	0.0	3.50	-4.05	12.34	5.882	7.334	356
2	.0764	0.0	.83	18.15	29.09	28.96	.4023	80.7
3	.0682	0.0	2.43	-2.06	2.499	1.378	1.528	80.7
4	.0682	3.22	0.0	15.62	18.81	20.45	1.904	52.1
5	.0455	-.37	0.0	-.90	.2199	.3419	.2044	16.3
6	.0336	-3.53	-.04	1.60	.2699	.9204	.7414	52.1
7 *	.0077	0.0	0.0	9.5	.7090	.7090	.0195	27.1

*) Without End Effector Data
**) Includes ALL Input Shaft Inertias Multiplied by the Square of the Gear Ration Between Input/Output

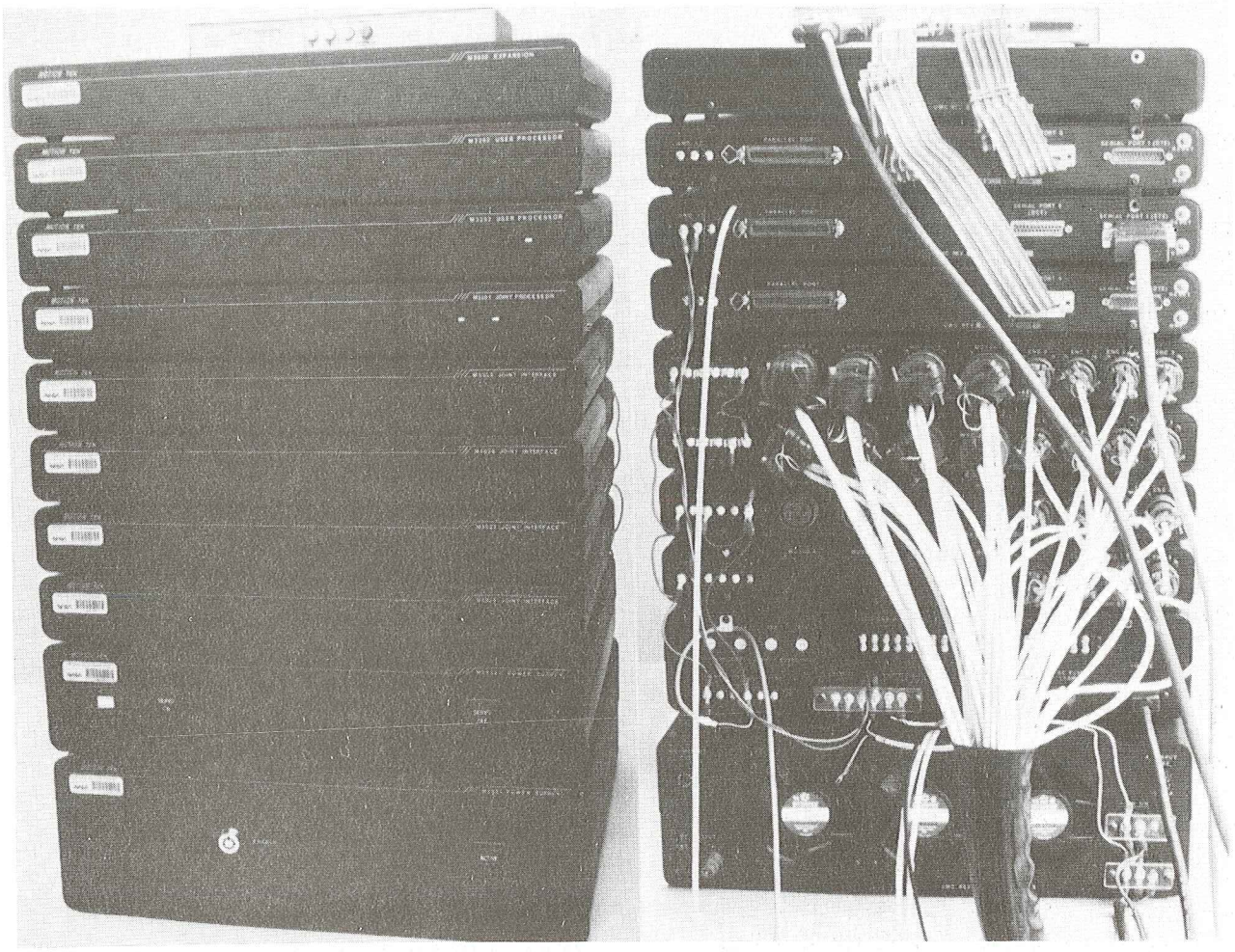


Figure 4. Eight D.O.F. ARM II Control Electronics, in Front and Rear Views

Table 3. Joint Natural Frequencies

Joint	Natural Freq. (Hz) w_a	w_b	Spring Constant (10^5 lb in/rad) A	B
1	6.9	10.0	9.0	18.8
2	6.9	10.0	9.0	18.8
3	6.9	9.9	2.5	5.15
4	6.9	9.9	2.5	5.15
5	8.7	12.7	1.1	2.34
6	11.1	27.1	0.48	1.20
7	8.7	12.7	1.1	2.34
8	27.1	39.5	1.1	2.34

Intel iSBX standard. The i/o bus makes the joint motion parameters memory mapped to the joint processor's address space. The joint interface card performs input data conversion and output control functions to the power amplifiers.

Each joint interface card has the following functions:

- 16 analog input channel A/D converter at a 12 bit accuracy.
- 4 optical encoder position counters.
- 4 digital tachometers.
- 4 digital control units for the PWM amplifiers.
- EEPROM non-volatile memory to store joint parameters.
- Watchdog timer.

The optical encoder interface is an up/down counter that can be used in 8 or 12 bit modes. The count has to be read periodically by the software to avoid more than one wraparound. The software computes the incremental change from one reading to the next and adds this change to a counter in memory.

The digital tachometers are devices that measure the time lapse from one positive edge in the input encoder pulse stream to the next. The software divides a constant with this time to get a quantity that is proportional to the joint velocity.

The PWM amplifiers in the UMC are controlled by digital signals that are generated on the joint interface cards. Each joint has two controlling registers, the motor is driven at a duty cycle that corresponds to the smaller value of the two. This arrangement ensures software control of the motor voltage even in case of component failures in the system. The amplifiers used to drive the AAI arm can deliver 20A of current at 60V each.

The PWM amplifiers in the UMC are unique in the sense that they do not have an integral current feedback loop. Conventional PWM circuits are strongly non linear. This necessitated the use of a feedback loop that linearizes the current transfer function of the amplifier. Such PWM amplifiers take an analog signal as input and produce a motor current on the output. The amplifiers in the UMC are designed such that the relationship between duty cycle and motor voltage, current and energy output is linear. This makes it unnecessary to have a feedback loop inside the amplifier. The output can be controlled directly with a digital signal, eliminating an intermediate analog stage. It is also because of this that the amplifiers produce an inherent velocity damping due to the tachometer effect of the motor. This reduces the magnitude of additional velocity damping needed. In such a system the motor itself is used as a tachogenerator, without any additional hardware. More on the UMC can be found in Refs. 4 and 5.

Input filters are also implemented since they are needed because the sharp rising and falling edges of the PWM signals driving the motor generate high energy noise spikes on all of the incoming signals. Such spikes are relatively easy to filter out because they are narrow. The incoming digital signals are first filtered by an R-C low-pass filter, after which a four stage digital sampling filter eliminates all pulses that are narrower than 2 microseconds. The analog signals are R-C filtered once in the input filter section and a second time on the joint interface card.

Control Computations

The multiprocessor system presently contains a total of three processors. The computer hardware system, including communication to the control station computer mode (and to the VME bus at the LaRC installation) are shown in Figure 5. The computational functions are distributed among the three processors as follows:

- Remote communication, trajectory generation, Cartesian servo and harmonic motion generator.
- Inverse and forward kinematics, gravity compensation, smart hand interface and compliance.
- Joint servo

The remote communication consists of a packet exchange via the fiber optic link. The hand controller node transmits a packet to the robot node. This packet contains a mode byte, a six byte (one for each degree of freedom) relative motion command, a one byte grasping force command and a checksum. When the robot node receives this packet it replies with its own that contains the following: Currently active control mode, robot position in the task space and joint space, the forces on the end effector, the finger position and a checksum.

The trajectory generator receives the incremental motion commands from the communication and generates the desired joint or task space positions for the robot. In one of the joint modes this is simply an addition of the incoming command to the joint setpoint. In task mode the input matrix for the inverse kinematics has to be generated. This consists of the end effector tip position which is generated by simple accumulation and the end effector

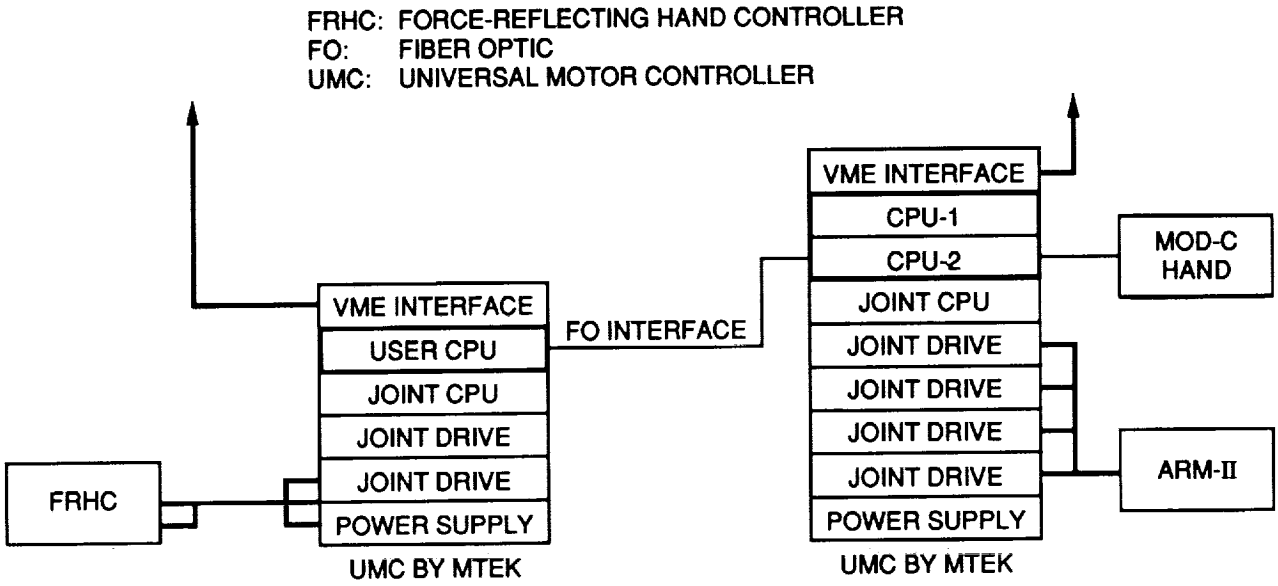


Figure 5. Eight D.O.F. ARM II Overall Control Schematics (Also Indicating VME Bus Interface at LaRC)

orientation matrix which is computed by consecutive rotations of this matrix while maintaining its orthonormality. When switching from one mode to another continuity of the robot position is maintained. This is accomplished by feeding the output of the forward kinematics into the input of the inverse kinematics while not in task mode.

The Cartesian servo improves trajectory tracking accuracy by establishing a servo loop in the task space. This servo loop compares the output of the forward kinematics to the desired Cartesian position and applies a correction to the input of the inverse kinematics to reduce the error.

The harmonic motion generator allows the robot to execute autonomous motions. These motions currently consist of straight line segments but curved segments will later be introduced. The tip velocity of the robot is controlled as a function of the position along the line of motion. This velocity has a sinusoidal profile. The compliance parameters can be independently preset for every motion segment. This allows the robot, for example, to autonomously track the inside edge of a hole or the outside envelop of an object. More on Cartesian Servo and Harmonic Motion Generator can be found in Refs. 6 and 7.

The inverse and forward kinematic computations are solved as an integrated package. The mathematics are based on the work performed in our group at JPL. (See Ref. 3). The governing principle in the computations is to use geometric reasoning to achieve optimum performance. One joint of the first four and one joint of the last four are parametrized, they maintain their positions from the last time they were moved in joint mode. The remaining six joints are computed based on the tip position and orientation requirement.

The gravity compensation precomputes the torque load of each joint based on the static weight of the links. This value is added to the feed forward field of the UMC joint servo. Such compensation improves the robot positioning accuracy substantially.

The smart hand interface controls the trimming values of the input channels of the smart hand force sensor, it controls the grasping and it reads the wrist force torque information. The forces and torques are converted to the laboratory frame and they are also low pass filtered. From the basic 500 Hz force readings a 100 and a 5 Hz force is generated. More on the Smart Hand can be found in Ref. 8.

The compliance function modifies the desired Cartesian position according to the forces detected. There are two types of compliance, spring and rate type. The spring type of compliance causes the robot position to be proportional to the force acting on it. This is equivalent to a spring. The integrating compliance causes that the velocity is proportional to the force on the tip. This is equivalent to a viscous damping. The two types of compliances can be mixed with each other as desired. More on compliance control can be found in Refs. 5 and 6.

The joint servo function is performed by a dedicated processor. This processor runs the code generator software. The code generator has a set of menues on which the user defines the robot being controlled. Based on this information the code generator writes highly optimal machine code that controls the robot. This makes it possible to switch polarities of various devices at runtime without changing the hardware, to move a joint from one output port to the next to facilitate debugging and to safely setup a new robot without risking damage to the hardware. This software also performs calibration at power on time to establish the robot absolute position accurately. Currently we cannot utilize the multiple redundancies of the robot fully because special software will have to be written that recognizes the failure of various devices and switches over to their alternative.

The control modes are the following:

- Freeze mode
- Neutral mode
- Joint - 1
- Joint - 2
- Task

Freeze mode means that the robot does not move and the brakes are all set. This is an alternative to the robot being completely turned off.

Neutral mode allows the robot to be moved by hand, it is gravity compensated but the control gains are set to 0.

The two joint modes allow the operator to control the joints using the hand controller. In joint-1 the upper and lower arm rotations are controlled, in join-2 the remaining six joints move.

Task mode controls the end effector position and orientation via the inverse kinematics. The current implementation freezes the joint 3 and 5 positions.

Advanced Bus, Advanced Processor

Due to the limited processing performance of our current system (1 MIPS per processor) and the limited transfer rate of our bus (MULTIBUS-I) we coded all of our computations in 32000 assembly language. The arithmetic is performed using binary fixed point instructions to gain execution time. Currently the control systems for the two (right and left) robots are not tightly coupled to each other. It is desirable for us to control both robots from the same tightly coupled environment and to increase our processing and bus communication performance. The only viable commercial alternative at hand would be a VME bus system. It is generally agreed upon by the research community today that the VME bus is no longer fast enough to support a high number of high performance processors working in a closely coupled environment. In fact, it is our view that the very concept of the bus for our application has to be rethought instead of just trying to build yet another bus that is a little faster then the previous ones.

For these reasons we came up with an advanced bus concept to support our new high performance multiprocessor environment. This system will have up to 16 processors on a bus with 10 or 15 MIPS of performance each. This new bus concept delivers about 10 times the performance of a VME bus at the same clock rate, and will be clocked at 20 MHz so we expect a better than 20 fold performance increase relative to VME bus systems. The processors will use fiber optic links as their primary means of communication outside the cardcage. Each of our processors will have 4 Mbytes of dynamic memory as well as a floating point processor and various i/o functions.

This new bus architecture brings several improvements in addition to the increased data processing capabilities. Since the arbitration on the bus is completely eliminated, the processors do not handshake to each other when the information is transferred. This makes it possible for several people to develop their software at separate locations and perhaps even using different programming languages. Once the programs are debugged they can be downloaded into the multiprocessor system and used. A number of processors can be running their software simultaneously while one of them is stopped and a new version of the software is downloaded and started. The program execution times do not change when a software piece is transferred from a single processor to the multiprocessor environment.

In a conventional bus system the bus load is not uniform. For example, if there is a common servo period there will be more traffic on the bus at the beginning and the end than in the middle in between. In a traditional bus every processor has to acquire control of the bus signals through arbitration before it can transfer data. This transfer could be either reading and writing. Typically a data item that is of global interest is generated once in a servo period and it is read several times by various different processors.

In our advanced bus the data is written into a FIFO register the time it is generated. The processor where the data is broadcast to all processors that need it simultaneously. Subsequently every processor will read its own copy as many times as needed without going to the bus. Since the transactions wait in their FIFOs till the bus can take them, bus overload cannot happen during peak traffic periods. The bus will only saturate if the long time average of the traffic exceeds the bus transfer capacity of 80 Mbytes/seconds. In comparison a VME bus will experience difficulties as soon as the momentary traffic exceeds about 4 Mb/s. The true bandwidth of a VME bus can only be utilized well if there is a single processor controlling the bus for an extended period of time.

The planned advanced bus and advanced processor will greatly facilitate the real-time (or, near-real-time) implementation of algorithms which are implied in the methodology that we have adopted for handling redundancy of the 8-D.O.F. AAI arms.

Methodology for Redundancy Handling

The proposed method for the inverse position transformation is based on parameterizing selected redundant joints in order to reduce the problem to a deterministic level. The solution is now based on the previously assigned, but adjustable values of parameters. For a class of robots for which arm decomposition is possible, arm redundancy can be distributed to individual subarms, such that a closed form of parameterized inverse kinematic solutions can be readily obtained from the subarm kinematics. The Null Space Manifold is then formed in the parameter space and characterized with an artificial potential field to represent manipulator internal as well as external behavior. The null space manifold can be easily scanned by varying the parameter values within their limits, and by checking the availability of the solution in the deterministic level. The artificial potential field over the null space manifold is formed based on a combination of several desired attributes such as the proximity to joint limits, the proximity to singularities, the proximity to current configuration, and the measure of static and dynamic manipulabilities.

The artificial potential field over the null space manifold can be used for the optimal adjustment of parameter values either automatically or manually by an operator. The automatic adjustment of parameter values is based on the local gradient of the potential field. Alternatively, a globally optimal joint trajectory can be formulated by analyzing the variation of the potential field at successive task points along the given task trajectory. The parameterization method also allows the visualization of manipulator internal performance through the display of a potential field in the parameter space. This provides a medium

for interactive interface between operator and manipulator for advanced teleoperation, through which the operator can decide whether, when and how to reconfigure the arm for optimal task execution. More on this methodology and related computational algorithms and techniques can be found in Ref. 3.

ACKNOWLEDGEMENT

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aerouantics and Space Administration.

REFERENCES

1. P. D. Spidaliere, "The Advanced Research Manipulator II: Design and Research Opportunities," to appear in Springer Verlag book "Robots with Redundancy," NATO Advanced Research Workshop, Salo, Italy, July 1988.

2. B. S. Thompson, et al., "An Experimental Investigation of an Articulated Robotic Manipulator with a Graphite Epoxy Composite Arm," Journal of Robotic Systems, Vol. 5, pp. 73-79, 1988.

3. S. Lee and A. K. Bejczy, "Redundant Arm Kinematic Control Based on Parametrization," Proc. 1991 IEEE Int'l Conf. on Robotics and Automation, Sacramento, CA, April 9-11, 1991.

4. A. K. Bejczy and Z. F. Szakaly, "Universal Computer Control System for Space Telerobots," Proc. 1987 IEEE Int'l Conf. on Robotics and Automation, Raligh, NC, March 31-August 3, 1987.

5. Bejczy, A. K., Szakaly, Z., Kim, W. S., "A Laboratory Breadboard System for Dual-Arm Teleoperation," Proc. of Third Annual Workshop on Space Operations, Automation and Robotics, JSC, Houston, TX, July 25-26, 1989, NASA Conf. Publication 3059, pp. 649-660.

6. Szakaly, Z. and Bejczy, A. K., "Performance Capabilities of a JPL Dual-Arm Advanced Teleoperations System," Proc. of SOAR'90 Workshop, Albuquerque, NM, June 26, 1990, pp. 30-41.

7. Bejczy, A. K., Szakaly, Z., "A Harmonic Motion Generator for Telerobotic Applications," Proc. of IEEE Intl. Conf. on Robotics and Automation, Sacramento, CA, April 9-11, 1991, pp. 2032-2039.

8. Bejczy, A. K., Szakaly, Z., Ohm, T., "Impact of End Effector Technology on Telemanipulation Performance," Proc. of Third Annual Workshop on Space Operations, Automation and Robotics, JSC, Houston, TX, July 25-27, 1989, NASA Conf. Publication 3059, pp. 429-440.

Performance Experiments with Alternative Advanced Teleoperator Control Modes for a Simulated Solar Maximum Satellite Repair

H. Das, H. Zak, W. S. Kim, A. K. Bejczy and P. S. Schenker
Robotics and Automation Section
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA

Abstract

This paper describes the experiments conducted at the JPL Advanced Teleoperator Laboratory recently to demonstrate and quantitatively evaluate the effectiveness of various teleoperator control modes in the performance of a simulated Solar Max Satellite Repair (SMSR) task. The SMSR was selected as a test task because it is very rich in performance capability requirements and it actually has been performed by two EVA astronauts in the Space Shuttle Bay in 1984. The main subtasks are: thermal blanket removal; installation of a hinge attachment for electrical panel opening; opening of electrical panel; removal of electrical connectors; relining of cable bundles; replacement of electrical panel; securing parts and cables; re-mate electrical connectors; closing of electrical panel; and reinstating thermal blanket. The current performance experiments are limited to thermal blanket cutting, electrical panel unbolting and handling electrical bundles and connectors. In one formal experiment seven different control modes were applied to the unbolting and reinsertion of electrical panel screws subtasks. The seven control modes are alternative combinations of manual position and rate control with force feedback and remote compliance referenced to force-torque sensor information. Force-torque sensor and end effector position data and task completion times were recorded for analysis and quantification of operator performance.

1. INTRODUCTION

In the last year, the Advanced Teleoperation (ATOP) Laboratory in the Robotics and Automation Section at JPL developed the ability to perform selected sub-tasks of a satellite repair operation with remotely operated manipulators. We chose the Solar Max Satellite Repair (SMSR) performed in an Extra-Vehicular Activity (EVA) by astronauts in 1984 as a model to demonstrate our capability. Our current capabilities of picking up tools for the repair operations, cutting taped seams of the thermal protection blanket around the Main Electronics Box (MEB) panel of a mock-up of the satellite, and removal and reinsertion of screws holding down the MEB panel enabled us to carry out a series of experiments to study operator performance with the alternative manual control modes and system parameters available on our system. A recent experiment we conducted and the results from it are reported in detail in this paper.

The history of operator performance evaluation with manual control modes in remote manipulation stretches back to the 1940's[14]. We have noted the lack of statistical rigor in much of the previous work in this field and results quoted have been mainly by visual inspection. This has been possible in simple experiments when differences are obvious in the data. It is widely accepted that position control without force feedback has faster completion times than both resolved motion rate control and joint rate control[10][11][15]. Means task completion times with pure position control were found to be 3-4 times better than with resolved motion rate control which, in turn, was 2-3 times better than joint rate control[11]. The advantages of resolved motion rate control were confirmed[6] for large workspace and limited manipulator speed and non-contact situations in task completion time. However, for small workspace applications, position control was found to be a better mode of operation. A recent study showed that position with shared compliant control [5] generated less interaction forces than position with force reflection on a telerobot. Performance measures used in the study were task completion times, cumulative forces of interaction and task board contact time. In this current work, statistical analysis has been applied rigorously to the data and we quantify the differences with probabilities that one manual control mode is better than another for specific performance measures.

2. ADVANCED TELEOPERATION SYSTEM

Detailed descriptions of components of the ATOP system have been reported elsewhere [1][2][12][13]. We briefly overview the important aspects of the system for completeness.

2.1 Master-Slave System

In the full configuration, the ATOP system is a dual-arm system [13]. We used a single arm (the right side of the system) in this study. On the master side is the 6 axes force reflecting hand controller. Switches on the FRHC allow the operator to activate and deactivate robot control. Incremental commanded motions of the FRHC are relayed to the slave. In manual position control mode, the operator is able to cover the larger workspace of the robot with the limited workspace of the FRHC by indexing. Inverse kinematics is not necessary on the FRHC side because hand controller joints approximate a cartesian coordinate frame and the operator automatically compensates for the small errors that occur.

A PUMA 560 robot with 6 degrees of freedom is used as the slave and a JPL Smart Hand [1] with a 6 axes force/torque sensor (F/TS) is used as the end effector. The 2-jaw gripper on the hand is used for holding the tool used in this experiment. Sensors on the hand also measure gripper jaw positions and forces. Local and remote sides are separated by a wall with one window that was kept obscured by a black curtain during the experiment.

2.2 Visual Feedback System

Six video display terminals are available for visual feedback to the operator in the full configuration of the ATOP system. However, only three were used in this experiment to display video images from cameras located in the remote side. The three camera image displays (identically used for all manual control modes in this study) are arranged beside each other in a console facing the operator: The terminal on the right shows the view from the camera placed to the right of the task board with a right side view of the task board; the terminal in the center shows the image from the overhead camera looking forward and down on the task board; and the terminal on the left shows the image from the camera facing the task board and from the rear of the robot. Two cameras, the camera at the back and on the right are mounted on pan and tilt units and can be controlled from the local (operator) side of the system. In ad-

dition, on those two cameras, zoom, focus and iris are also controllable from the operator side.

The telerobotic configuration editor (TCE) [9], an iconic interface used for selecting configuration parameters, setting gain values, and control modes, was programmed for the alternative control modes under study. It was used by the experimenter for quickly alternating between the various control modes during the experiment, as required by the randomization of the tasks presented in our experiment. The TCE interface was repositioned so that it was not directly visible to the subjects during the experiment.

2.3 Operator Control Station

The configuration of the operator control station is as shown on Figure 2.1. The three upper displays were not used by the subjects during the experiment. The subjects were seated to the left of the hand controller and faced the console. Two foot switches on the floor were used to control the tool, one for unbolting and the other for bolting the screw. Lighting in the control room was turned off for better visibility of the display terminals.

2.4 Task Board and Tools

The task board consists of a mock-up of three sides of the Solar Maximum Satellite consisting of 5 panels, the MEB and its 4 adjacent panels. The MEB panel, on-loan from NASA Goddard Space Flight Center, is an accurately scaled flight replica of the panel used on the satellite. The remaining panels and the frame holding all the panels were built in-house. Thermal protection blanket, consisting of gold Mylar film on a thin foam sheet, covered the mock-up. In this experiment, the MEB panel face is exposed, simulating the completion of the first phase of the satellite repair procedure of cutting and folding back of the blanket. The configuration of the task board and the relative location of the side and front view cameras are shown in Figure 2.2.

The tool used in the experiment was a modified power screwdriver operated by the subjects with foot switches located in front of the operator seat in the control room. The tool has a handle attachment to provide a firm grip in the jaws of the JPL Smart Hand. While not in use the tool is held on the tool caddy with Velcro tape.

3. EXPERIMENTS

Three operator performance experiments have been conducted to-date on the ATOP system. In the first experiment, a comparison between manual position control with force reflection and manual position control without force reflection as alternative modes of control of the master-slave system was performed. The SMSR sub-tasks of tool pick-up, thermal protection blanket tape cutting, and MEB panel screw removal were the tasks performed in the experiment[3]. In the second experiment[4], the same sub-tasks were performed with visual feedback images from three black-and-white (B+W) cameras placed in the remote site (displayed on three monitors in the control room) compared to operator performance with an image from a stereo-pair of B+W cameras at the remote site (displayed on a single monitor). The most recent in the series of SMSR experiments in the ATOP has been the comparison between seven alternative manual control modes in the performance of the screw unbolting and bolting of a MEB panel screw and in the remainder of this paper, this experiment is described in detail.

3.1. Task

The task selected was socket head screw removal and reinsertion. The steps in the execution of the task are:

1) move in a straight line forward (1.25" in the positive x direction) to bring the tool bit to the head of the socket screw.

2) engage the tool bit with the screw head.

3) unscrew by activating of the *unscrew* foot switch while withdrawing the tool as the screw unbolts.

4) withdrawal to approximately the start position after the screw was free of the threaded part of the screw hole,

5) return to the location of the hole.

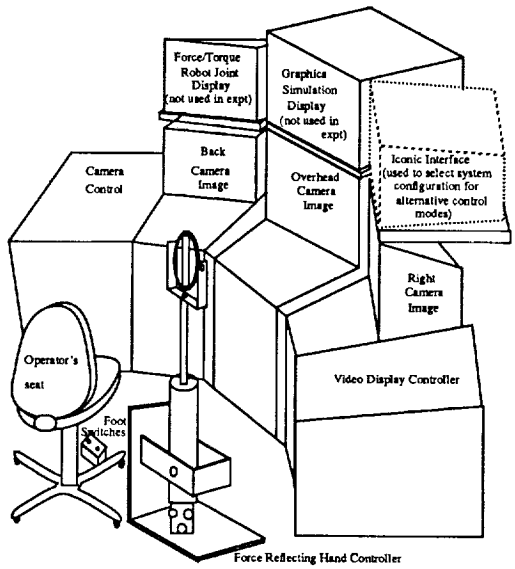


Figure 2.1 Operator Control Station

6) reinsertion of the screw by activation of the *screw* foot switch.

During the withdrawal phase the screw remained attached to the magnetized bit of the tool unless external forces or motion of the tool unseated it.

3.2 Experiment Design

A completely randomized single factor, within subject (repeated measures) design was specified for this experiment. The single factor represents the independent variable "control modes", and its seven levels correspond to the seven control modes described in Section 4. Seven subjects participated in the study. The presentation order of the seven control modes was randomized for each subject. Each control mode was presented three times, and so in all, every subject had to perform the task 21 times (i.e. 3 repetitions times 7 control modes). Seven dependent measures, described in detail in the following section, were defined on each subject. A training procedure, described in Section 4.1, was planned to reduce within-subject and between-subject variance without an extended training period.

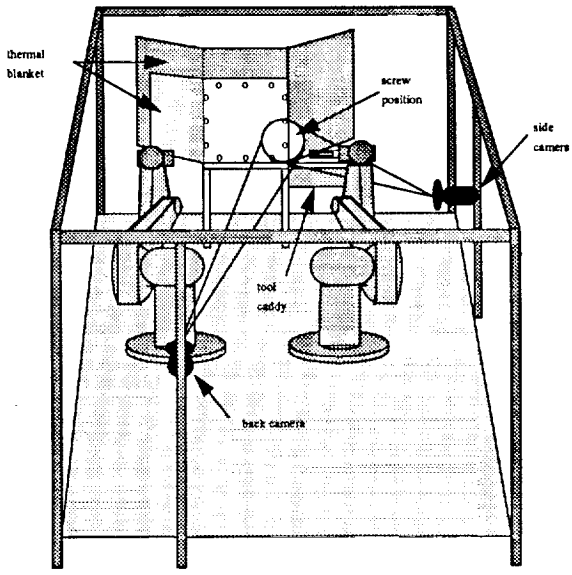


Figure 2.2 Remote Environment

3.3 Dependent Variables

The seven dependent measures in the experiment were:

1. Average Completion Time, the mean time it took to successfully complete the task.
2. Average Force, the mean force exerted by the end effector on its environment during the contact phase alone.
3. Average Torque, the mean torque exerted by the end effector on its environment, during the contact phase alone. The Average Force or Torque value is computed using

$$AverageForce = \frac{\sum_{i=1}^{N} f_i}{N}$$

where N = number of data samples in the contact phase, f_i was the magnitude of i^{th} force or torque vector. The force magnitude is defined as

$$f_i = \sqrt{f_{xi}^2 + f_{yi}^2 + f_{zi}^2}$$

where f_{ni} was the n^{th} component axis *(x,y,z)* for the i^{th} sample of force. The equation for the torque is similar, replacing the translational axes *(x,y,z)* with the rotational ones *(roll, pitch, yaw)*.

4. Cumulative Task Force, the time interval summation over the contact time of the forces exerted by the end effector on its environment multiplied by the sampling interval.
5. Cumulative Task Torque, the time interval summation over the contact time of the torques exerted by the end effector on its environment multiplied by the sampling interval. The equation for the Cumulative Task Force and Cumulative Task Torque measures goes as follows:

$$CumulativeTaskForce = \sum_{i=1}^{N} f_i \Delta t$$

where N= number of data samples in the contact phase, f_i is the magnitude of the i^{th} force or torque vector, Δt *sec* = the sampling interval. Cumulative Task Torque is computed similarly.

6. Number of Errors, the errors were defined as either a drop of the screw, or having to "recover" more than twice during task execution. A recovery was defined as a re-engagement followed by turning of the screw after accidental loss of engagement.
7. Subjective ratings of control modes by the subjects based on their collective experience with the alternative control modes. The score was obtained by having the subjects indicate on a scale of 1 to 9 their rating of each control mode with 9 being the best and 1 the worst.

The first five dependent measures above have historically been used in the ATOP laboratory to evaluate operator performance. Although it is not clear if these measures are the best way to characterize operator performance, intuitively, they do measure variables of interest and a cost can be attached to poor performance according to these variables in space applications; time spent in space is very expensive and excessive force/torque exertion can damage delicate instruments. Also, task completion time has been used by all previous researchers in the field so it should serve as a measure for comparison of our results with those quoted in the literature.

The cumulative force/torque measures combine task interaction forces/torques with contact time. It is possible and it would be interesting to combine the respective elements in different weightings to but that would require an experiment designed for that purpose and we have not elected to perform such an experiment yet.

3.4 Manual Control Modes

All control modes described below are closed-loop due to visual feedback to the operator. However, the terms open-loop and closed-loop have been used to indicate whether kinesthetic force feedback to the operator is used for manual control.

3.4.1 Position Control without Force Reflection or Compliance (PNA)

This is essentially an open-loop manual control mode in which the operator commands robot position relying only on visual feedback, and not on a kinesthetic force feedback via the FRHC. There is no force reflection or compliance added to the system as shown in Figure 3.1 It should be noted that although the stiffness of the robot is large when maintaining a commanded position, it is not infinite due to the finite control gains implemented on the robot PD joint controllers and saturation limits of joint actuators.

3.4.2 Position Control with Force/Torque Sensor (F/TS)-Based Force Reflection (PWF)

Here is a closed-loop manual control mode whereby measurements from the F/TS on the robot hand sampled at 1KHz are used to drive the FRHC. The control architecture is illustrated on Figure 3.2. .

3.4.3 Position Control with Remote Side Compliance (PWC)

In this open-loop mode of manual control, position commands from the FRHC are used to drive the slave robot. However, compliance implemented on the robot modifies the operator commanded positions to the robot causing it to yield to task interaction forces. The low pass filtered force and torque control loop in the remote site emulates a damped spring connected to the robot hand for each cartesian axis. The slave manipulator is thus made compliant to external forces and torques. This so-called shared compliance control [7] architecture is shown on Figure 3.3.

3.4.4 Position Control with Position Error Based Force Reflection (PEF)

The PEF control mode is described in [8]. The essential feature of this closed-loop mode is the implementation of compliance on the slave robot, and using the position error, generated during any force interaction with the task due to the compliance, to drive the FRHC. Position errors are transformed to kinesthetic feedback to the operator. Figure 3.4 below illustrates the method. It was found that this control architecture was able to generate much greater force reflection ratios than those used in the F/TS based force reflection mode (PWF), while maintaining system stability.

3.4.5 Rate Control without Force Reflection or Compliance (RNA)

This control mode is rate without force reflection or compliance as shown on Figure 3.5. This is the traditional resolved motion rate control mode used extensively in manual control.

3.4.6 Rate Control with F/TS-Based Force Reflection (RWF)

In this control mode, illustrated in Figure 3.6, force/torque measurements from the sensor on the robot hand were added to the spring/damper force implemented for manual rate control. With this control mode, close to the home position of the FRHC, subjects could feel forces/torques corresponding to sensor readings but as they moved away from the home position, the spring force returning the FRHC to its home position would become significant enough to prevent discrimination between the spring force and the reflection force. This is a combination that has not been attempted elsewhere. The force reflection gains were identical to those used in the position control with F/TS force reflection. A local side PD controller is used to implement the spring/damper effect common to rate control input devices.

3.4.7 Rate Control with Remote Side Compliance (RWC)

Remote side compliance with identical gains to those used for position control with remote compliance was implemented in this control mode. The control architecture was as shown on Figure 3.7. The subjects had to rely on visual feedback alone for performing the task in this control

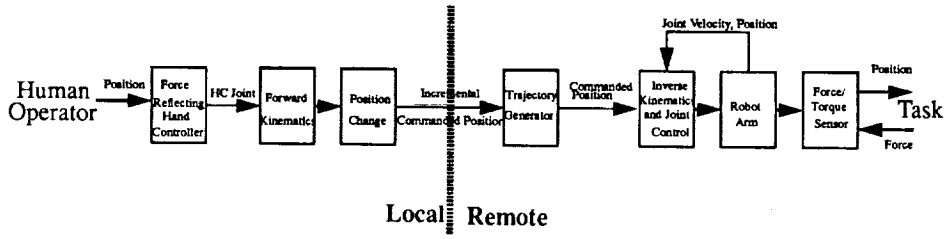


Figure 3.1 Position Control without Force Reflection or Remote Side Compliance

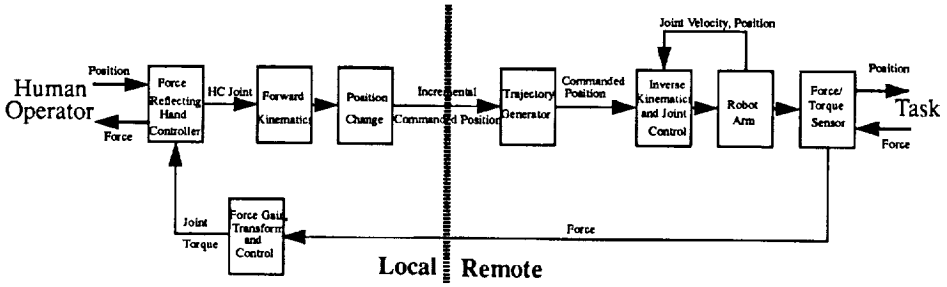


Figure 3.2 Position Control with F/TS Based Force Reflection

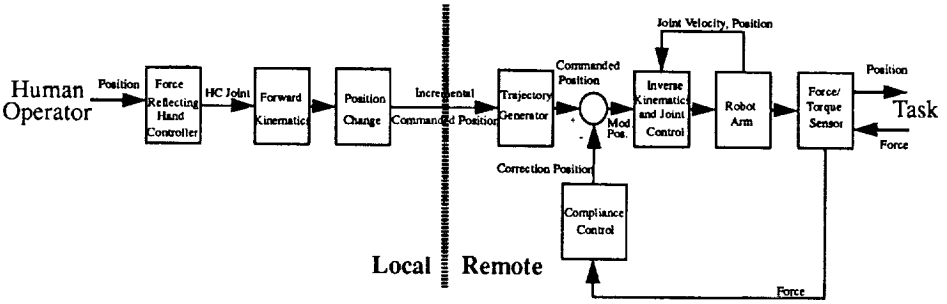


Figure 3.3 Position Control with Remote Side Compliance

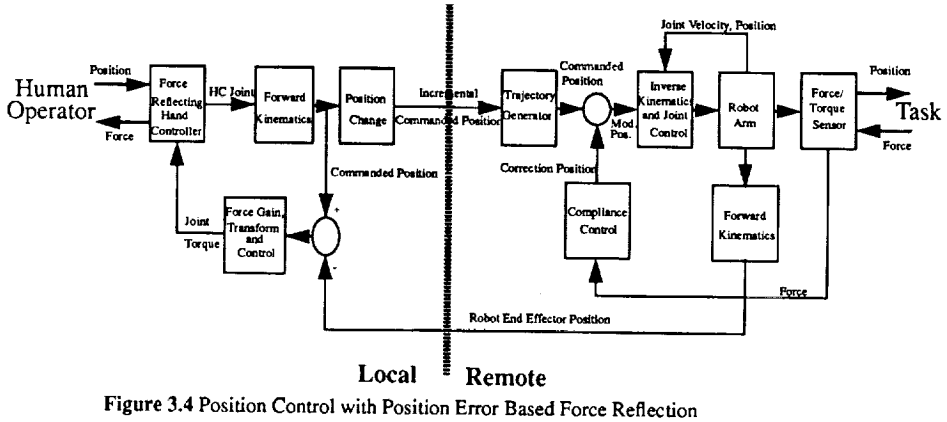


Figure 3.4 Position Control with Position Error Based Force Reflection

mode.

4. EXPERIMENT PROCEDURE

4.1 Subject Training

Teleoperation, by its nature, is a complex operation, requiring the timely integration of many sensory, cognitive, and motor inputs and functions by the operator, in order to achieve a satisfactory level of performance. Proficiency in teleoperation is thus largely a function of training. We believe that there is evidence to support hypothesis that the training period may vary (according to task complexity) from a few days to weeks or even months before a leveling off of the performance curve occurs.

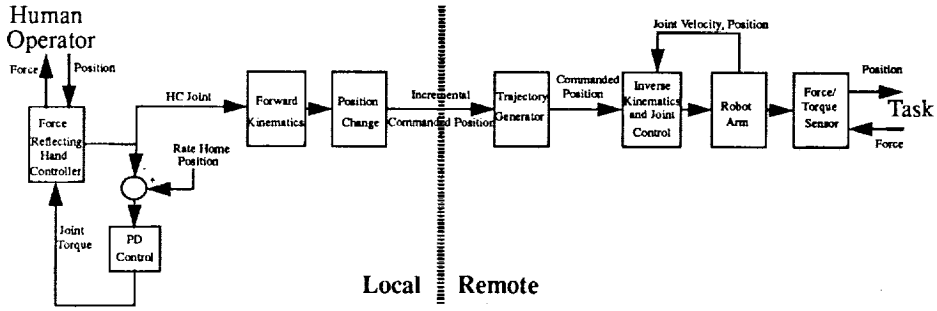


Figure 3.5 Rate Control without Force Reflection or Remote Side Compliance

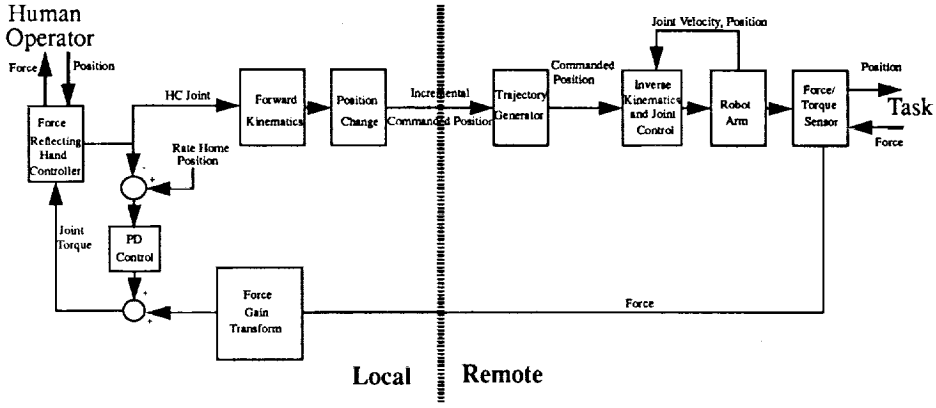


Figure 3.6 Rate Control with Force/Torque Sensor Based Force Reflection

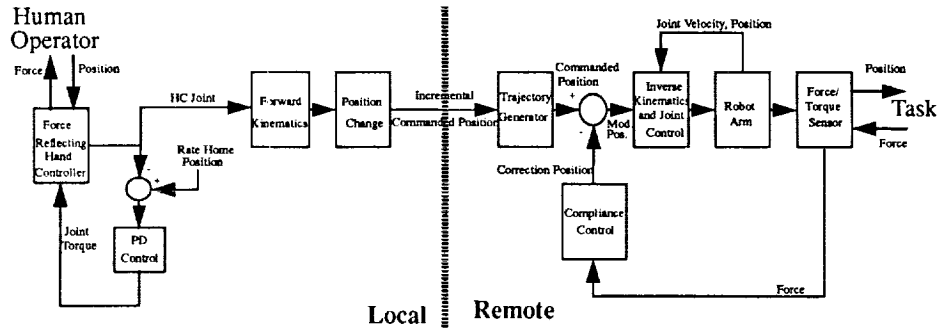


Figure 3.7 Rate Control with Remote Side Compliance

In this experiment, due to time constraints, instead of training subjects until they reached their level of proficiency, a criterion was established to denote the end of an individual's training period. This criterion stated that an individual had reached the end of his/her training whenever the ratio of the standard deviation to the mean of performance for five consecutive trials was 0.15 or less. By doing so we managed to provide all subjects with comparable training that resulted in reduced within-subject variance and between-subject variance.

Since in most cases several days had elapsed between the completion of training and the onset of the experimental session, subjects were required to perform the task successfully once for each of the control modes at the beginning of the experimental session. The data from this practice run was not used in the evaluation of performance.

4.2 Data Collection

Data collected during the experiment were: *x*, *y*, *z*, *pitch*, *yaw* and *roll* positions of the robot end effector in a world coordinate frame, *x*, *y*, and *z* forces and *pitch*, *yaw* and *roll* torques sensed by the F/TS resolved to a world coordinate frame, and end effector gripper jaw positions and forces. Each variable collected was sampled at 1000Hz then averaged over 16 samples before being stored in real-time during the experiment on a personal computer (PC) dedicated for data collection. Each stored value of a variable thus represented the average value of the respective variable over 16 milliseconds.

4.3 The Experimental Run

The chronology of an experimental session was as follows:

1. The subject was instructed on the procedure and the objectives. Instructions given to the subject at the start of the experiment were:
   a. Perform the task as quickly as you can.
   b. Try to exert minimal force and torque.
   c. Avoid committing errors.
2. The subject was given a practice run with each of the seven control modes in succession to re-familiarize himself with the modes.
3. The subject was asked to read the rating questionnaire to be filled after the end of the session.
4. The experiment was started and data was collected for each run. The

subject was informed of the control mode he was about to operate with prior to each run, so that he would know what to expect. The experimenter started each experimental run with a countdown to synchronize the subjects's start with the manual activation of the data collection program. The task was judged complete by the experimenter when the tool bit stalled at the end of the screw reinsertion phase at which time the data collection program was stopped. Three main functions performed by the two experimenters during the experiment were:

a. manual activation and deactivation of the computerized data collection system at the start and the end for each run by pressing a key on the computer dedicated for data collection,

b. re-configuration of the teleoperation system for the next run, and

c. observation and manual recording of any errors and other exceptional events occurring during each run.

5. At the end of the session, the subject was asked to fill out the questionnaire rating the different modes.

The experimental sessions including rest periods lasted about 2 hours for each subject.

5. RESULTS

Table 5.1 contains the means and standard deviations of subjects' performance for the seven measures: the completion time, the average force and torque, the cumulative contact force and torque, the no. of errors, and the subjective ratings of the control modes. Three major hypotheses were tested for statistical significance using the Multivariate ANOVA procedures [16]. However, visual inspection of the data plots can offer a wealth of valuable qualitative observations. These are not subjected to rigorous statistical testing in order to keep the experimentwise error (experimentwise α) [16] from inflating.

5.1 Visual Inspection of Data

Means and standard deviations for the respective performance measures are plotted on Figure 5.1a-g and listed on Table 5.1. Ranking the seven

Table 5.1: Means (Standard Deviations)

control mode	average time	average force	average torque	cum. task force	cum. task torque	no. of errors	subject ratings
PEF	57.62 (5.82)	3.03 (0.32)	0.83 (0.13)	13.47 (6.78)	3.82 (2.32)	0.43 (0.79)	6.86 (1.35)
PWC	67.05 (11.04)	6.25 (2.30)	1.28 (0.43)	127.83 (105.17)	23.86 (14.50)	0.43 (0.54)	6.14 (0.69)
PWF	70.80 (19.87)	7.79 (1.87)	2.05 (0.41)	207.47 (88.61)	53.62 (21.11)	0.43 (0.79)	6.71 (1.11)
PNA	75.40 (15.07)	10.37 (2.12)	2.76 (0.55)	342.01 (97.10)	88.46 (27.02)	0.14 (0.39)	5.00 (1.00)
RWF	101.36 (15.57)	12.91 (3.44)	2.59 (0.44)	574.66 (110.82)	117.99 (25.43)	0.57 (1.13)	3.86 (0.90)
RWC	108.94 (67.36)	10.41 (2.98)	1.44 (0.28)	385.52 (245.39)	59.63 (44.09)	0.86 (1.46)	3.57 (1.13)
RNA	144.12 (49.80)	15.59 (2.56)	2.62 (0.42)	1086.94 (382.30)	198.05 (104.17)	0.86 (0.69)	2.29 (1.25)

control modes based on the data plots results in the PEF being always the best followed by PWC, while RNA is always the worst, with RWF being the next worst. In the middle, PWF, PNA, and RWC interchange their rank as a function of the measure being looked at.

Standard deviations are greatly reduced with the PEF condition in all five measures, but generally they are comparable for all control modes, in terms of both average force and torque measures. This is not the case for completion time, where in the rate modes (excluding the RWF) the variability in performance was much larger than in the position modes. The cumulative task force and torque plots also indicate that there exists a larger variability of performance in the rate modes than in the position modes.

Compliance appears to be better than the F/TS force reflection in both rate and position modes, in terms of the force/torque indices, however compliance does not appear reduce completion time.

RWC was the only rate mode capable of matching and even outperform-

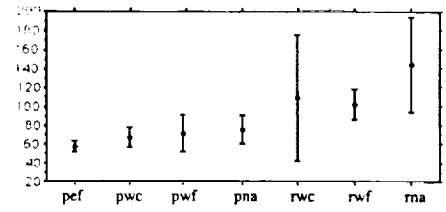


Figure a Task Completion Time

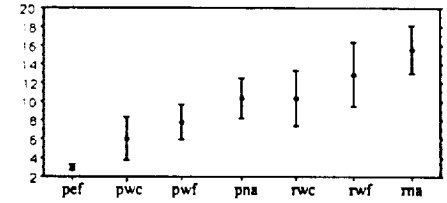


Figure b Average Force (lbs)

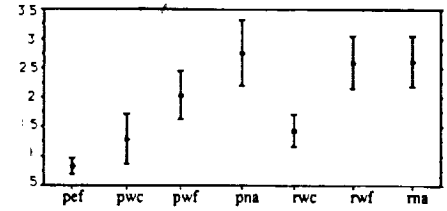


Figure c Average Torque (ft.lbs)

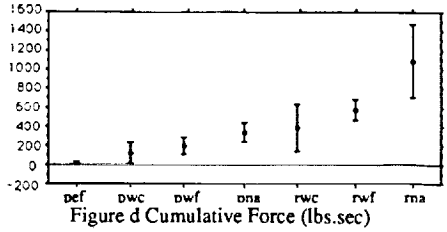


Figure d Cumulative Force (lbs.sec)

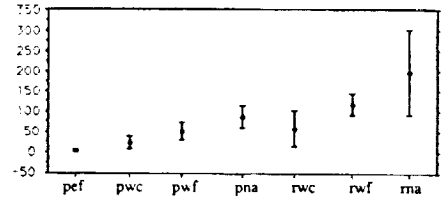


Figure e Cumulative Torque (ft.lbs.sec)

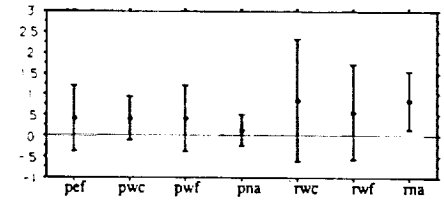


Figure f No. of errors

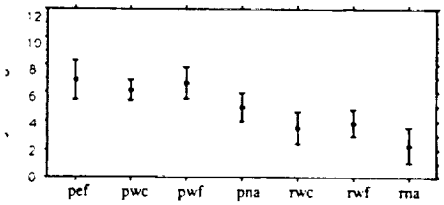


Figure g Subject Ratings

Figure 5.1 Means and Std. Dev. for Respective Dependent Measures

ing some of the position modes, albeit only with respect to the force and torque measures.

Inspection of the remaining data plots (Figures 5.1f, g) reveals that the number of errors did not prove itself as a particularly sensitive measure of performance, mainly due to the fact that there was great variability in performance for that measure, in all the control modes.

On the other hand, subject ratings clearly indicates a general preference of position modes over all rate modes. Ranking of the control modes based on subject ratings yielded the PEF as the favorite control mode, with PWF and PWC following closely. PNA was the least favorite among all position modes. The order of preference for rate control was RWF, followed closely by RWC and RNA. The fact that subjects preferred the F/TS force reflection modes over the compliance modes, contrary to the ranking of these modes by other performance measures (average force and torque, cumulative task force and torque) suggests that the "objective" performance measures may not be adequate in some sense. An explanation may be the loss of operator control (the remote manipulator has a semi-autonomous behavior in its response to external forces/torques) when compliance is implemented and the subjects may have preferred the feeling of being-in-control even at the cost of poor performance.

5.2 Hypotheses Testing

Results of the analysis are summarized in Table 5.2. The following three hypotheses were formulated for testing with the Multivariate ANOVA (evaluating the Wilks' Lambda Statistic, R), on the first three performance measures:

1. The first hypothesis tested that there was no difference in the means of performance between the position and the rate control modes. Analysis shows that R=7.63, with probability, p=0.04, a significant difference between the means of performance in the position modes and the rate modes, in favor of the position control.
2. The second hypothesis tested was that the newly implemented control mode, PEF, was not better than the best among all other position modes, PWC. Testing resulted in R=6.56 and p=0.05, a significant difference in favor of the PEF control mode.
3. The third hypothesis compared the two traditional control modes, i.e. pure position control with no force reflection or compliance (PNA) against pure rate control (RNA). Analysis yielded R=7.88, with p=0.04, a significant difference in favor of the position control mode.

Table 5.2 Multivariate ANOVA Hypothesis Test Results

HYPOTHESIS	MULTIVARIATE ANOVA WILKS' LAMBDA (R)	DF	P
1. Position vs. Rate	7.63	3,4	0.04
2. PEF vs. PWC	6.56	3,4	0.05
3. PNA vs. RNA	7.88	3,4	0.04

6. CONCLUSION

In this paper, we have demonstrated several points:

1. Position control modes yield better overall teleoperation performance than rate control modes, and were preferred by operators. We should note that training, which took a significantly longer time than actual experiment time, probably affected the perception of the subjects on their preferences. We have not analyzed the training data to discern trends similar to those obtain from the experiment.
2. Position error based force reflection with compliance implemented on the remote side is the best of all control modes in this study. This finding can be attributed to improved force reflection ratio alone. However, disadvantage of this mode is that the feel of the FRHC is sluggish and force feedback is slighlty delayed due to the very limited bandwidth of the force reflection.
3. If one were to choose between operating with the pure position control mode and the pure rate control mode in situations were contact and dexterity are the main requirements of performance, there is little doubt that pure position control is preferred.

In addition we have observed the advantage of compliance over the F/TS force reflection in terms of improved force/torque management performance, although subjects indicated in their subjective ratings a slight preference for the force reflection modes. However the advantage of FT/S force reflection over the pure position and rate modes is apparent in terms of most measures.

While our conclusions mostly apply to the selected sub-task of the SMSR, screw removal and re-insertion, under our experimental conditions and for our performance indices, we do believe that they can serve as a guide for recommendations in similar real or simulated teleoperation situations.

7. ACKNOWLEDGEMENT

This work was carried out at the Jet Propulsion Laboratory under contract with National Aeronautics and Space Administration.

8. REFERENCES

[1] A. K. Bejczy. Smart hand: Manipulator control through sensory feedback. Technical Report D-107, Jet Propulsion lab., Pasadena CA, January 1983.

[2] A. K. Bejczy, Z. F. Szakaly, and W. S. Kim. A laboratory breadboard system for dual-arm teleoperation. In *SOAR '89 Workshop*, NASA Johnson Space Center, Houston TX, July 1989.

[3] H. Das, H. Zak, and P. Lee. Operator Performance with and without Force Feedback on SMR Sub-tasks with the Advanced Teleoperator System, JPL Robotics and Automation Section Interoffice memorandum No. 3470-91-011. Jan., 1991.

[4] D. B. Diner. Monocular TV versus Stereo TV for Solar Maximum Repair Sub-tasks, JPL Robotics and Automation Section Interoffice Memorandum, (in preparation).

[5] W. S. Kim, P. G. Bakes, S. Hayati, and E. Bokor. Orbital replacement unit changeout experiments with a telerobot testbed system. In *IEEE International Conf. on Robotics and Automation*, pages 2026-2031, Sacramento, CA, April 1991.

[6] W. S. Kim, S. R. Ellis, M. E. Tyler, and L. W. Stark. A comparison of position and rate control for telemanipulators with consideration of manipulator system dynamics. *IEEE Journal of Robotics and Automation*, RA-3(5), October 1987.

[7] W. S. Kim, B. Hannaford, and A. K. Bejczy. Shared Compliant Control for Time-Delayed Telemanipulation, In *First Symp. on Measurement and Control in Robotics*, NASA Johnson Space Center, Houston, TX, June 1990.

[8] W. S. Kim. A new scheme of force-reflecting control. In *SOAR '91 Workshop*, Houston TX, July 1991.

[9] P. Lee, A. K. Bejczy, P. Schenker, and B. Hannaford. Telerobot configuration editor. In *IEEE International Conf. on Systems, Man, and Cybernetics*, Los Angeles, CA, November 1990.

[10] D. Mullen. An evaluation of resolved motion rate control for remote manipulators. MIT Draper Lab. Report No. T-562, May 1972.

[11] J. L. Nevins, T. B. Sheridan, D. E. Whitney, and A. E. Woodin. The multi-moded remote manipulator system.In *Remotely Manned Systems*, ed. E. Herr, California Institute of Technology, Pasadena CA, pages 173-187, 1973.

[12] P. S. Schenker, A. K. Bejczy, W. S. Kim and S-K. Lee Advanced Man-machine Interfaces and Control Architecture for Dexterous Teleoperation. To be presented at *IEEE Oceans '91*, Honolulu, HI, Oct. 1991.

[13] Z. F. Szakaly, and A. K. Bejczy. Performance capabilities of a JPL dual-arm advanced teleoperation system. In *SOAR '90 Workshop*, Albuquerque NM, July 1990.

[14] A. Tustin. The nature of the operator's response in Manual Control and its Implications for Controller Design, J. IEEE, v91, part IIa, n. 2, 1947.

[15] D. R. Wilt, D. L. Pieper, A. S. Frank, and G. G. Glen. An evaluation of control modes in high gain manipulator systems. *Mechanism and Machine Theory*, 12(5), pages 373-386, 1977.

[16] J. A. Woodward, D. G. Bonett, and M-L. Brecht. Introduction to Linear Models and Experiment Design, pub. Hardcourt, Brace and Javanovich, San Diego, 1990.

EFFECTS OF SPATIALLY DISPLACED FEEDBACK ON REMOTE MANIPULATION TASKS

Meera K. Manahan, Mark A. Stuart, John M. Bierschwale, Ellen Y. Hwang, and A.J. Legendre*
Lockheed Engineering and Sciences Company
*NASA Johnson Space Center
Houston, Texas 77058

ABSTRACT

Several studies have been performed to determine the effects on computer and direct manipulation task performance when viewing conditions are spatially displaced. Whether results from these studies can be directly applied to remote manipulation tasks is questionable. The objective of this evaluation was to determine the effects of reversed, inverted, and inverted/reversed views on remote manipulation task performance using two 3-Degree of Freedom (DOF) hand controllers and a replica position hand controller.

Results showed that trials using the inverted viewing condition showed the worst performance, followed by the inverted/ reversed view and the reversed view when using the 2x3 DOF. However, these differences were not significant. The inverted and inverted/reversed viewing conditions were significantly worse than the normal and reversed viewing conditions when using the Kraft Replica.

A second evaluation was conducted in which additional trials were performed with each viewing condition to determine the long term effects of spatially displaced views on task performance for the hand controllers. Results of the second evaluation indicated that there was more of a difference in performance between the perturbed viewing conditions and the normal viewing condition with the Kraft Replica than with the 2x3 DOF.

INTRODUCTION

Telerobotics will play an essential role in the assembly, operation, and maintenance of NASA's existing and future spacecraft. Direct views of the worksite will not be available or sufficient for many telerobotic tasks, so cameras will provide the primary mode of visual feedback. Due to structural and logistical constraints on Space Station *Freedom*, cameras cannot always be located to provide the optimal view of the task site, and the views presented to the operators will most likely be spatially displaced. Task performance with distorted views may be adversely affected, with some distortions causing more of a decrement than others.

Camera views are generally referenced to the equipment performing the task. During direct manipulation (when tasks are performed using hands or simple tools), a "normal" camera view, with no spatial displacement, is considered to be a view from behind the person's hands. With remote manipulation (in which operations are performed by mechanical devices controlled by a human from a distance), a "normal" view is considered to be from behind the manipulator arm.

Various types of spatial displacements have been classified and examined in relation to task performance. Spatially displaced feedback can take on four different forms: (1) *angular displacement*, in which the reference point of the camera in relation to

the arm is either displaced along the horizontal axis or the vertical axis, (2) *reversed displacement*, in which the camera faces the task board and the arm, (3) *inverted displacement*, in which the camera is behind an upside-down with respect to the manipulator arm, and (4) *inverted-reversed displacement*, in which the camera is upside-down and faces the task board and the manipulator arm. Performance using visual perturbations is usually compared to performance using a direct view or a normal camera view.

Several studies have been performed to determine the effects on computer and direct manipulation task performance when the viewing conditions are spatially displaced (Bernotat, 1970; Kim, Tendick, and Stark, 1987; Smith and Smith, 1962). Whether results from these studies can be directly applied to the remote manipulation tasks to be performed on various space platforms is questionable.

There have been few studies examining the effects of spatially displaced feedback on remote manipulation task performance. A study conducted by Stuart and Smith (1989) investigated the effects of normal, reversed, inverted and inverted/reversed viewing conditions on a remote manipulation task. Their results found that remote manipulation performance using the inverted viewing condition was significantly worse than performance with any of the other views, and that the direct view provided the best performance. A follow-up study showed that all perturbed viewing conditions were significantly worse than the normal view, but none of the perturbed views was significantly better or worse than another (Stuart, Bierschwale, Sampaio and Legendre, 1990).

The two remote manipulation evaluations mentioned above used a replica mini-master position hand controller system. Since these studies were completed, an extensive investigation performed by several laboratories at the Johnson Space Center has shown that task performance with orthogonal rate hand controllers (in particular, with the 2x3 DOF type hand controllers) is significantly enhanced when compared to task performance with replica mini-master hand controllers (NASA/MSD, 1991). Based on the results of this study, the baseline hand controller for Space Station Freedom was deemed to have a 2x3 DOF rate configuration. In light of these results, it was considered imperative to incorporate the orthogonal baseline configuration into current investigations of perturbed visual feedback on remote manipulation task performance.

This experiment augmented the previously mentioned perturbed feedback remote manipulation evaluations, to include the 2x3 DOF hand controller configuration. It was hypothesized that performance using perturbed viewing conditions would vary between the orthogonal and the replica hand controller, and that strategies for task performance may also differ. The objectives of this evaluation were as follows:

1. Determine the effects of reversed, inverted, and inverted/reversed views on remote manipulation task performance with a 2x3 DOF hand controller.

2. Compare the effects of spatially displaced views on remote manipulation with a 2x3 DOF hand controller and the Kraft Replica hand controller.

METHOD

The primary objectives of this evaluation consisted of determining the effects of spatially displaced views on task performance with the 2x3 DOF and comparing it to the Kraft Replica.

Subjects

The four test subjects from the previous study (Stuart, Bierschwale, Sampaio, and Legendre, 1990) also participated in this study so that a valid comparison could be made. All test subjects were experienced in performing remote manipulation tasks and with both hand controllers used.

Apparatus

Testing was conducted in the Remote Operator Interaction Laboratory (ROIL) at NASA's Johnson Space Center.

A Honeywell Apollo modified 2x3 DOF, orthogonal, rate mode hand controller and a Kraft Telerobotics force-reflecting, replica, mini-master hand controller were used to operate a Kraft Telerobotics 6 DOF remote manipulator to perform the task. A single closed circuit color camera and a 19" color monitor were used to provide the camera view at the subject's workstation.

Variables

The independent variables in this evaluation were the hand controllers (Honeywell 2x3 DOF and Kraft Replica), viewing conditions (normal, reversed, inverted, and inverted/reversed), trials (three), and subtasks (four). This evaluation used a nested repeated measures design; all subjects were exposed to all levels of the independent variables used. However, each hand controller's data were analyzed separately due to operational differences.

The dependent variables in this evaluation were trial completion time, subtask completion time, number of errors, and subjective questionnaire responses.

Procedure

Test subjects were given an overview of the remote operation and instructed to complete the task as quickly and accurately as possible. No information on camera position was given.

The task performed in this investigation was functionally similar to multi-axis translation and alignment tasks which will be performed by the telerobots on space platforms. Three trials of the task were performed with each viewing condition, and the task consisted of four subtasks.

To familiarize test subjects with the task, three trials of the remote operation were first performed with the direct view, in which no camera view was provided. During this time, the test administrator coached subjects on general techniques to avoid manipulator joint limits by using cues on the manipulator and keeping the end effector within a defined work zone. After completion of three trials using the direct view, three trials were performed with each of the four camera viewing conditions in counterbalanced order. Hand controller order was also counterbalanced.

RESULTS

Performance and subjective measures were analyzed using the Clear Lake Research Analysis of Variance (CLR ANOVA) statistics program and Duncan's multiple range test.

ANOVAs were used to analyze the following variables: task completion times across all trials (i.e., if one viewing condition was significantly longer than another for all three trials), task completion times within each trial (i.e., if one viewing condition was significantly faster than another during any one of the three trials), and completion time within each subtask (i.e., if the any of the viewing conditions were significantly different from another viewing condition during any of the four subtasks). Separate ANOVAs were performed on data collected from the 2x3 DOF and the Kraft Replica hand controllers. Results from analyses performed on subtask completion times and errors were complementary to the other analyses, and thus will not be presented for the sake of brevity. Only significant results will be presented in the sections below.

Completion Times for 2x3 DOF Hand Controller

Marginally significant differences (p=.0711) were found during the analysis of the viewing condition completion time data. Figure 1 shows the completion times for the trials with each viewing condition. Note that trials using the inverted viewing condition were longer than trials using any of the other three viewing conditions. The completion time for the three trials was significantly different (p<.05), with the Duncan's paired-comparison test revealing that task completion time for trial 1 was significantly longer than completion time for trials 2 and 3.

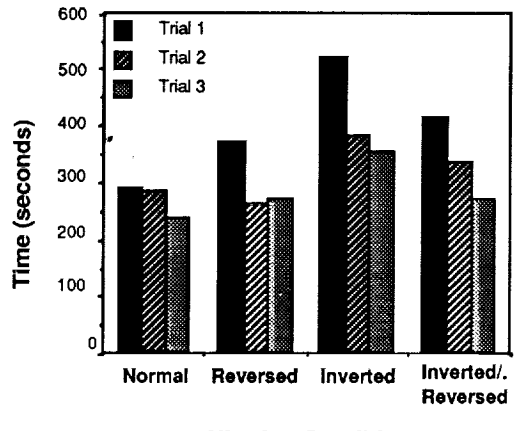


Figure 1. Completion Times for 2x3 DOF Hand Controller

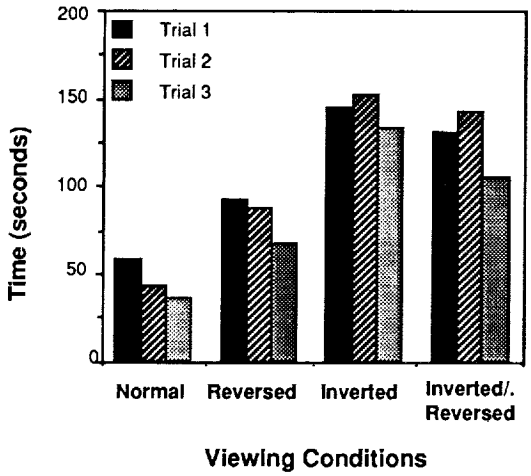


Figure 2. Completion Times for Kraft Replica Hand Controller

The ANOVA performed within each of the three trials showed that the viewing conditions were significantly different only during trial 3 ($p<.05$). The Duncan's test revealed that the inverted viewing condition was significantly slower than the other three viewing conditions.

Completion Times for Kraft Replica Hand Controller

There was a significant difference among the four viewing conditions ($p<.01$). The Duncan's pairwise comparison test indicated that the normal and reverse viewing conditions were both significantly quicker than the inverted and inverted/reversed viewing conditions. Figure 2 shows the completion times for each of the viewing conditions and trials.

An ANOVA performed within each trial showed that the viewing conditions were significantly different within each of the three trials ($p=<.05$ for trial 1, $p<.05$ for trial 2, and $p<.05$ for trial 3). Duncan's tests indicated that the normal viewing condition was significantly faster than the inverted and inverted/reversed viewing conditions during all three trials and that the reversed viewing condition was significantly faster than the inverted viewing condition, only during trial 3.

Trials with the Kraft Replica were quicker than trials with the 2x3 DOF hand controller because of the difference in control system processing time between the two hand controller configurations.

Analyses of Subjective Responses for 2x3 DOF Hand Controller

Analysis for the issue of mental workload revealed marginal significance ($p=.0636$), with the inverted viewing condition requiring more mental workload than the other three viewing conditions. Subjects rated the normal and inverted/reversed viewing conditions as being significantly more acceptable than the reversed and inverted for executing right and left movements ($p<.05$). Subjects rated the normal, reversed, and inverted/reversed viewing conditions significantly better than the inverted viewing condition for overall acceptability, with the normal condition rating significantly more acceptable than the inverted/reversed viewing condition ($p<.01$).

Analyses of Subjective Responses for Kraft Replica Hand Controller

Subjects rated the normal viewing condition as requiring significantly less mental workload than the other three viewing conditions. The reversed viewing condition was also rated as requiring significantly less workload than the inverted viewing condition ($p<.01$). When subjects were asked if their discomfort affected performance, they believed that the inverted condition affected performance significantly more than the

normal and reversed viewing conditions ($p<.05$). The inverted view was found to be the worst for movements in all three axes (up/down, right/left, and in/out). The normal view was found to be significantly more acceptable overall than the inverted and inverted/reversed viewing conditions. In addition, the reversed view was significantly more acceptable than the inverted viewing condition ($p<.01$).

DISCUSSION

The first objective of this evaluation was to determine the effects of reversed, inverted and inverted/reversed views on remote manipulation performance with a 2x3 DOF hand controller. Results showed that while inverted views often produced the slowest times, the only significant difference between the viewing conditions was during the third trial. Trials 2 and 3 were both significantly faster than trial 1, suggesting that a significant amount of learning occurred between trials 1 and 2. Learning may have occurred within each trial, with the largest differences between viewing conditions occurring during the first two subtasks.

Results suggest the same degree of learning occurred at trials 1 and 2 for all viewing conditions, but less learning occurred with the inverted viewing condition than with the other viewing conditions from trial 2 to trial 3. There were no differences among the normal, reversed and inverted/reversed views at trial 3. Subjects may have been able to adapt more readily to the reversed and inverted/reversed views by trial 3.

Even though left and right were transposed in the reversed view, it was often easier to use than the normal view. The views from the camera located behind the manipulator, the normal and inverted views, were partially occluded by the manipulator. This is an unfortunate consequence of placing a camera behind the object that will be used to perform the task. The reversed view provided a complete display with no occlusion from the manipulator.

The inverted view proved to be the most difficult to use, because it not only provided perturbed visual feedback, but also was occluded by the arm. In fact, trials with the inverted viewing condition had the highest average number of unsuccessful grapple attempts with both hand controllers. This was most likely due to the occlusion of the second task piece by the manipulator. Subjects could not determine the specific location of the second task piece, but grappled to test the position of the grippers.

Subjective data for the 2x3 DOF configuration was consistent with performance data: subjects rated the inverted viewing condition significantly lower in overall acceptability over the other three viewing conditions. There was no difference between the normal view and the reversed view, indicating that a reversed view was equally as effective for performing this task.

The second objective of this evaluation was to compare effects of spatially displaced views on remote manipulation with the 2x3 DOF configuration and the Kraft Replica. Results from data collected with the Kraft Replica showed the same order for performance with viewing conditions as the 2x3 DOF, with the normal view being the easiest to use, followed by the reversed view, the inverted/reversed view and the inverted view. Successive trials with the normal and reversed views were shorter in length, where as trial 2 was longer than trial 1 for the inverted and inverted/reversed views. Therefore learning may have been more erratic for the inverted and inverted/reversed views with the Kraft Replica while learning was more constant across viewing conditions with the 2x3 DOF. The normal view was found to be significantly better than the inverted and inverted/reversed views for all three trials, and the reversed view was found to be better than the inverted during the third trial. This suggests that some degree of learning occurred at trial 3 for the reversed viewing condition, but significant amounts of learning did not occur with the inverted and inverted/reversed viewing conditions.

Subjective data for the Kraft Replica showed that the normal view was significantly better than all other viewing conditions. Subjective data for the 2x3 DOF showed that the inverted view was significantly worse than all the other views, and that the normal

view was better than reversed view. These subjective impressions are consistent with task performance, in which trials with the inverted view were significantly longer than trials using the normal view.

From these results, it was hypothesized that the two hand controllers displayed different learning curves. Thus, data was collected to determine the learning effects of successive trials on task performance. The apparatus and procedure followed were the same, with the following exceptions: (1) three new subjects performed six trials of the task, instead of three, (2) subjects practiced trials with the normal view, and only counterbalanced data collected for the perturbed viewing conditions were statistically analyzed, and (3) subjective responses were not collected.

EVALUATION OF LEARNING EFFECTS

Separate ANOVAs were performed on data collected from the 2x3 DOF and the Kraft Replica hand controllers. ANOVAs were performed on data collected with only the perturbed views. Data analysis was not conducted on the normal view completion times because the normal view was not counterbalanced with the perturbed views.

Three separate ANOVAs were performed on completion times for each hand controller. The ANOVAs looked at the following: task completion times across all trials (i.e, if one viewing condition was significantly longer than another for all three trials), task completion times within each trial (i.e, if one viewing condition was significantly faster than another during any one of the three trials), and completion time within each subtask (i.e., if the any of the viewing conditions were significantly different from another viewing condition during any of the four subtasks). Results from analyses performed on subtask completion times were complementary to the other analyses, and thus will not be presented for the sake of brevity. Only significant results will be discussed.

Data collected using the 2x3 DOF showed that there was no difference among the perturbed viewing conditions, but there was a significant difference among the six trials ($p<.05$). The Duncan's test showed that trials 4, 5 and 6 are significantly faster than trials 2 and 3. The learning curves for the six trials are shown in Figure 3. The normal and inverted/reversed viewing conditions seemed more erratic in the first two trials, but quickly became constant. All perturbed viewing conditions appeared to level at trial 4 and remain fairly stable for trials afterward. During trials 4, 5, and 6, the inverted and reversed viewing conditions performed somewhat faster than the normal view, probably because the normal view was the first condition to be presented, for practice.

Results showed that there was no significant difference among the perturbed viewing conditions, but there was a significant difference among the trials ($p<.05$), with the trial 6 being significantly faster than trials 1, 2, 3, and 4. A graph showing the learning curves for each viewing condition with the Kraft Replica is presented in Figure 4. The graph shows that trial completion time for the inverted and inverted/reversed viewing conditions were erratic for all trials. Times for the reversed viewing condition had begun a constant trend toward trial completion time with the normal view.

Data from the 2x3 DOF showed that trials using the inverted/reversed view were longer than other views, but not significantly longer. Trial completion times for all perturbed views also appeared to stabilize during trials 5 and 6, and trials with the inverted and reversed views were somewhat quicker than trials with the normal view.

Data collected using the Kraft Replica showed that all perturbed views took longer than the normal view, even though trials with the normal view were performed first. Trials using the inverted and inverted/reversed views were erratic for all six trials, but trial completion times for the reversed view had begun to stabilize by trials 4, 5, and 6. A comparison of the two hand controllers showed that there was a greater difference between the perturbed views and the normal view when using the Kraft Replica. There was little decrement in

performance between the perturbed views and the normal view with the 2x3 DOF.

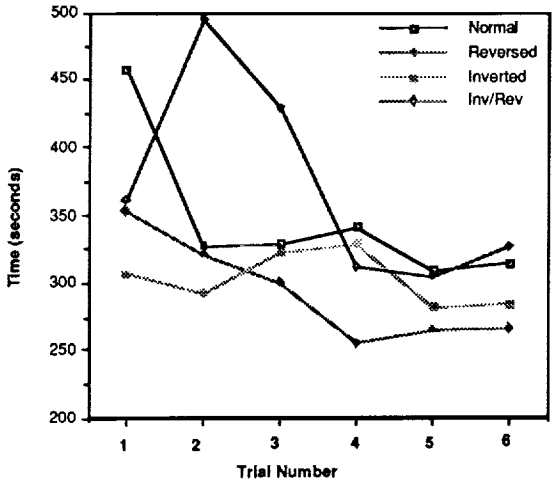


Figure 3. Completion Times for 2x3 DOF Hand Controller (Normal View for Comparison Only)

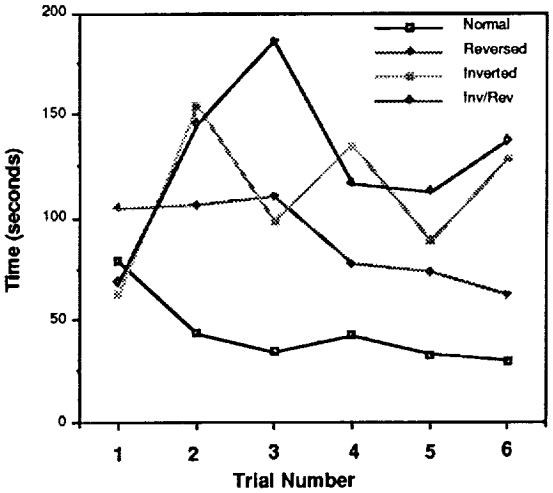


Figure 4. Completion Times for Kraft Replica Hand Controller (Normal View for Comparison Only)

Trials with the perturbed viewing conditions stabilized more with the 2x3 DOF than with the Kraft Replica over the six trials. The inverted and inverted/reversed views were more erratic than the reversed view for both hand controllers, and the reversed view was often quicker than the other perturbed views. Data collected from both hand controllers suggests that learning occurs more steadily with the 2x3 DOF hand controller. Learning may occur quicker with the 2x3 DOF configuration, although there is a limit to the amount of improvement from the initial trial. The processing time of the 2x3 DOF control system configuration limits the speed at which inputs are induced to the manipulator, and this limit may restrict the amount of improvement possible from an early trial to a later trial.

CONCLUSIONS

The purpose of this investigation was to determine the effects of spatially displaced feedback on remote manipulation task performance with two hand controllers. Studies conducted previously had examined spatially displaced feedback with the Kraft Replica hand controller. Since the baseline hand controller for Space Station *Freedom* was deemed to be a 2x3 DOF configuration, it was concluded that studies examining spatially displaced feedback should incorporate 2x3 DOFs.

Across both controllers, the inverted and inverted/reversed viewing conditions were generally more difficult to use than the normal and reversed viewing conditions. Tasks could be performed with the reversed view with little or no decrement in task performance. The reversed view often provided better performance than the normal view, because the normal view could be blocked by the manipulator performing the task.

There were no significant differences among viewing conditions with the 2x3 DOF. The inverted and inverted/reversed views were worse than the normal view with the Kraft Replica. Results also showed that learning was more constant and predictable with the 2x3 DOF configuration. Learning was more erratic with the Kraft Replica, possibly because it was difficult to recover from incorrect inputs. There was more cross-coupling with the Kraft Replica as well, and this hand controller may be more difficult to maneuver with accuracy and precision. This erratic behavior with the Kraft Replica may explain an anomaly in an earlier study performed by Stuart, et.al (1990).

The scope of this study was to examine the use of 2x3 DOF hand controllers for performing tasks with spatially displaced feedback. A host of other considerations need to be evaluated before the issues surrounding perturbed visual feedback can be resolved as listed below :

1. Use of perturbed visual feedback while performing various tasks.

2. Use of perturbed visual feedback when several camera views are provided to perform a task. Generally, more than one view of the task board will be available on space platforms during tasks. If several views are provided, operators may rely less on the perturbed views. If the perturbed views are absolutely necessary for completion of the task, then performance may be degraded.

3. Various types of spatial displacements. The most probable displacements that operators on *Freedom* will have to experience will be angular displacements, either along the vertical or the horizontal axes. These types of displacements need to be studied with tasks similar to those to be performed in space.

4. Investigation of changes in the control mode of the hand controller to be compatible to the camera, where the point of reference would move with respect to the camera. This conversion would place the mental transformation of the perturbed view on the control system, instead of the operator.

REFERENCES

Bernotat, R.K. (1970). *Rotation of Visual Reference Systems and Its Influence on Control Quality*. IEEE Transactions on Man-Machine Systems, Volume MMA-11, No. 2, Page 129.

Kim, W.S., Tendick, F., and Stark L.W. (1987). Visual Enhancements in Pick-and-Place Tasks: Human Operators Controlling a Simulated Cylindrical Manipulator. In *IEEE Journal of Robotics and Automation*, Volume RA-3, No. 5, Page 418.

National Aeronautics and Space Administration (1991). *Space Station Hand Controller Commonality Test Report.* (NASA Report JSC-32125). Man Systems Division.

Smith, K.U., and Smith, W.M. (1962). *Perception and Motion.* W.B. Saunders and Company: Philadelphia.

Stuart, M.A., and Smith, R.L. (1989). *Spatially Displaced Visual Feedback and the Performance of FTS-like Tasks* (NASA Tech. Report JSC-23547). Houston, Texas: NASA Lyndon B. Johnson Space Center.

Stuart, M.A, Bierschwale, J.M., Sampaio, C.E, and Legendre, A.J. (1990). *The Role of Perturbed Sensory Feedback in Space Remote Manipulation Tasks -- Preliminary Guidelines* (NASA Tech. Report JSC-24643). Houston, Texas: NASA Lyndon B. Johnson Space Center.

N93-11967

Importance of Numerosity and Distribution of Articulations Within the Digits, Wrists, and Arms of Telemanipulators Confronted with Dextrous Assembly Tasks

Steven F. Wiker, Neil Duffie, and Thomas Yen

Wisconsin Center for Space Automation and Robotics
1357 University Avenue
University of Wisconsin, Madison 53706

Abstract

This study was conducted to evaluate the impact of successive exclusion of degrees-of-freedom within digits (thumb and index finger) and wrist when performing tasks demanding varying degrees of end effector kinematic complexity. The analysis was performed by successively constraining articulations in the fingers (thumb and index) and wrists or human subjects via splints, and then timing performance of part movement, positioning, and assembly operations requiring different levels of kinematic freedom. Analysis of performance times showed that transferring degrees-of-freedom, or articulations, from the digits to the wrist and arm had little or no material affect upon manipulation performance times if wrist and arm kinematics were unrestricted. If wrist and arm kinematics are not constrained and are of high-quality, our findings show that transferring kinematic freedom from the digits to the wrists and arms or remote end effectors offers limited or no consequence in terms of remote assembly operations. The importance of such findings, in terms of development and implementation of commercially-attractive telemanipulators, is discussed.

N93-11968

ABSTRACT FOR SOAR 1991

The Development of System Components to Provide Proprioceptive and Tactile Information to the Human for Future Telepresence Systems

Ammon K. Wright, Captain, USAF Det 1 Armstrong Laboratory, BBA
Wright-Patterson AFB, Ohio 45433-6573 (513) 255-3671

This paper presents system components that are being implemented to augment teleoperated systems by providing both force and tactile information to the human operator. The concept proposed is the control of a manipulator to perform tasks; i.e. flight line maintenance and repair of combat aircraft or satellites while under the control of a human operator at a remote location to maintain mission effectiveness in a hostile environment.

The human would control the motion of the manipulator via a master system with information from the remote site being fed back by direct stimulation of the humans sensory mechanisms or by graphic interpretation of displays. We are interested in providing the operator feedback of position, force, auditory, vision, and tactile information to aide in the human's cognitive ability to control the manipulator. This sensory information from the remote site would then be presented to the operator in such a manner as to enhance his performance while providing him a sense of being present at the remote location, this is known as telepresence.

This paper also discusses the research done by the Human Sensory Feedback (HSF) facility at the Armstrong Laboratory to provide tactile and proprioceptive feedback to the operator. The system components of this system include tactile sensors and stimulators, dexterous robotic hands, and the control of positioning and operating industrial robots with exoskeletal mechanisms.

Session A5: TELEROBOTICS AND MECHANISMS

Session Chair: Prof. Mark Friedman

ADVANCED MECHANISMS FOR ROBOTICS

JOHN M. VRANISH

NASA/Goddared Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

An overview of applied research and development at the Goddard Space Flight Center (GSFC) on mechanisms and the collision avoidance skin for robots is presented. First the work on robot end effectors is outlined, followed by a brief discussion on robot-friendly payload latching mechanisms and compliant joints. This, in turn, is followed by the collision avoidance/management skin and the GSFC research on magnetostrictive direct drive motors. Finally, a new project, the artificial muscle, is introduced. Each of the devices is described in sufficient detail to permit a basic understanding of its purpose, fundamental principles of operation, and capabilities. In addition, the development status of each is reported along with descriptions of breadboards and prototypes and their test results. In each case, the implications of the research for commercialisation is discussed. The chronology of the presentation will give a clear idea of both the evolution of the R&D in recent years and its likely direction in the future.

INTRODUCTION

In recent years NASA has launched an extensive effort in robotics for space applications; the idea being to assist and augment manned space flight and, perhaps equally important, to give a major boost to U.S. industrial competitiveness by transferring the technology into the private sector. In this effort, research on mechanisms, actuators and motors and their associated sensors and controls, has traditionally been sacrificed on the altar of a rush for more and more "compute power", sophisticated graphical and animation packages and controls techniques. This, of course, results in lopsided progress; twenty first century computers and controls coexisting side-by-side with nineteenth century mechanisms, actuators and motors. But; robotics is a systems challenge and if one portion of the system is limited, the system is limited. A balanced approach to R&D is a must if real systems progress is to be made. The intent of this paper is to show what is being done at GSFC to bring about this balance and, as a by-product, the benefits and opportunities open to U.S. industry.

Accordingly, an overview of applied research and development at the Goddard Space Flight Center (GSFC) on mechanisms and the collision avoidance skin for robots is presented. First the work on robot end effectors is outlined, followed by a brief discussion on robot-friendly payload latching mechanisms and compliant joints. This, in turn, is followed by the collision avoidance/management skin and the GSFC research on magnetostrictive direct drive motors. Finally, a new project, the artificial muscle, is introduced. Each of the devices is described sufficiently to permit a basic understanding of its purpose, capabilities and operating fundamentals. In addition, the development status of each is reported along with descriptions of breadboards and prototypes and their test results. In each case, the implications for commercialisation is discussed. The chronology of the presentation will give a clear idea of both the evolution of the R&D in recent years and its likely direction in the future.

I. END EFFECTORS

In this section, GSFC end effector R&D is discussed. The GSFC "Gripper-Nut Runner" (fig. 1) is described first, followed by a new emerging concept based on "Spline Screws".

The "Gripper-Nut Runner" has been under development at GSFC for more than three years and has evolved to the point shown in Fig.1. Since space is a "micro-g" environment, objects must be fastened to something (say, for example, Space Station) to prevent their drifting away. The function of the "Gripper-Nut Runner", therefore, is to grasp a dedicated interface attached to such an object using the gripper and to use the nut runner to loosen the fastener which fixes it to Space Station. This permits the robot to grasp objects and unfasten them and to move and re-attach them. This system has been proven in the GSFC robotics lab as well as several other NASA-affiliated activities around the country. It is rugged and durable and has repeatedly withstood forces at the finger tips on the order of
200 lbf. Of its component subsystems, three have commercial possibilities.

a. The "Split-Rail" Parallel Gripper [1]. This device(the gripper portion of the system shown in Fig. 1), was designed and patented by a NASA engineer, and has obvious commercial possibilities as a general purpose industrial gripper. This is a high performance wide throw parallel action gripper that uses a unique "split-rail" concept to make it simple, light, and compact (hence, inexpensive to manufacture). It is made primarily of anodized aluminium with straight machining cuts (no grinding) and rolls on cylindrical bearings throughout its stroke and so is strong, precise, responsive and will not jam under side loads.

b. Very Large Scale Integration (VLSI) gripper controller [2]. This state-of-the art gripper controller was developed as part of a NASA Small Business Innovative Research contract (SBIR). It is a dual-axis system that incorporates unique and innovative custom Application Specific Integrated Circuits (ASPICS) to permit unusual performance in terms of filtering, precision and smoothness of motor control. It has a unique and compact power supply system to permit its being mounted on the gripper. Interfaces, hardware and software are kept serial and simple. This system is fully developed and functional. It has obvious commercial implications for any type of servo joint. For example, it can readily be adapted, in a modular fashion, to form the basis for an entire robot control system.

c. Rolling Friction Fingers [3]. These simple and compact gripper fingers (Fig. 1) provide a vectored, rolling friction guidance and locking system between the robot gripper and the dedicated interface. Invented and designed in-house at GSFC, these fingers enable the gripper to acquire a dedicated interface of an object despite large initial misalignments and to guide the gripper to a seat and lock with the object, providing low friction and smooth operation throughout, despite the presence of large side forces. It permits the gripper to release the object under strong side loads, a very important safety hedge against the object being caught and jammed in the jaws. Also, because of its rolling action, it cannot scar or "burr" the object regardless of the forces. And, because of this rolling action, force feedback sensing is cleaner and the size of the motors required to drive the gripper screw can be reduced. Most of the GSFC grippers are equipped with this device. It has proven out in the laboratory over continued use; in one instance, for example, permitting a weak gripper motor to release an object under side loads of 100 lbf. and consistently making it easier for robots to acquire and grasp objects using both passive and/or active compliance. This should have a wide range of commercial applications wherever guidance and latching and locking of objects is required under conditions of large misalignments and opposing forces.

d. Wrist-Driven Auto Changer [4]. Invented and developed in-house at GSFC, this simple, compact and strong device provides a safe and reliable means for space robots to exchange end effectors. It is actuated by a unique simple camming action which occurs between mechanisms in a tool interface plate, a keying element in the tool storage holster and another mechanism attached to the robot as the robot stores or removes the tool. The entire system is

passive with all action being initiated by the robot actuators and controls. Hence, many of the sensors of the robot joints can be used (to include the encoders or resolvers and the wrist 6 vector force feedback sensor) to aid in monitoring the status of and controlling the Auto Changer. Also, many of the interfaces to robot control are already inherently taken care of. The device is inherently safe and reliable because one of the prime sources of Auto Change mishap, inadvertent release, is impossible. It has been tested extensively in the GSFC robot lab and has met or exceed every expectation. It has even been operated in a teleoperator mode without force feedback-a very difficult feat. This should have a limited; but significant niche' market in the commercial world.

GSFC is in the process of developing an entirely new approach to end effectors based on a simple straight-forward concept called "Spline Screws" [5]. This concept was invented and developed in-house at GSFC and represents an entire space fastening strategy of which end effectors is but one portion. We will develop the explanation of this concept by a simple example (fig. 3). We assume that the robot wrist has a dual roll capability about a common center. The inner roll terminates in a splined screw driver and the outer roll provides the means to rotate the object being grasped. This constitutes our end effector. The object to be moved is pierced by a screw (typically 0.5 in. dia.) which is splined so as to mate with the splined screw driver on the one side and the fixture to which the object will be attached on the other. The place of attachment has a small rotating fixture which is splined to cooperate with the piercing screw of the object on the side away from the gripper. We will begin by assuming the object is fastened to an attachment point. The robot would position the end effector over the splines of the screw piercing the object. The robot would guide and seat the inner roll splined screw driver and then guide and seat the outer roll torque tabs into slots in the top of the object. The splined screw driver would be turned clockwise. The splines of the screw driver would mesh and lock with those of the object screw, the object screw would turn with the rotating fixture. In the process, the object screw would translate towards the attachment fixture. This would unlock it from the fixture and lock the object to the end effector. The end effector would be free to leave with the object and maneuver it pending attachment to another fixture. Once aligned with and seated on this new fixture, the end effector would be turned counter clockwise and the object screwed off the end effector and onto the new fixture. The object would be either attached to the end effector or the attachment fixture or both at all times.

The splines permit the end effector screw driver to capture the object screw yet are sufficiently course that cross-threading between the two would be impossible. The same would be true of the interface between the object screw and the attachment fixture. On the other hand, the object screw would be captive in the object so it could have a very fine thread with a short lead and be lubricated to give great holding forces with modest amounts of torque (approx. 600 lbf from 8 ft-lbf torque typical). The system would not backdrive so brakes would not be necessary on the screw driver and an attached object could survive launch forces. It is also clear that foot prints on the object and the attachment fixture would be very small (on the order of 1.25 in dia.) as would the end effectors. The system would be basic, simple, strong and very reliable. It would also be extremely versatile.

To demonstrate the versatility of the "Spline Screw" technique the example of an Auto Change is given. In this case we attach the screw to a common tool interface by means of slightly compliant wavy springs. But where the nut was the pierced object in Fig. 2, it is an electrical connector in the Auto Change which is partially trapped so that it can move in translation only with respect to the common tool interface. Thus, as the screw is turned, the electrical connector nut would translate upwards, inserting the pins into the end effector electrical connector and releasing the Auto Change and its tool from its holster. At this point, the connector nut would be stopped by the upper portion of the Auto Change. But; the screw would still be turning so it, in turn, would translate downwards towards the tool, compressing

the lower wavy spring washer in the process. In doing so, however, The Auto Change and end effector would seat and lock together. The process of storing tools would be the reverse. The screw would be turned counterclockwise, the nut would translate down until the tool locked to the holster. In the process the tool and electrical pins would separate from the end effector. Torques required would be modest (approx. 8 ft-lbf.) and the system would seat and lock or store with authority and certainty. And, of course, the dimensions and weights would be minimal.

II. ROBOT-FRIENDLY PAYLOAD LATCHING MECHANISMS.

It appears the most efficient robot-friendly payload latching mechanism to date would simply use the "Spline Screw" approach in yet another configuration. The design is closely related to the Auto Change described above; with a few minor additions.

III. COMPLIANT JOINTS.

Compliance is a critical component in the interaction between the robot and the object being grasped. Without it, a robot cannot capture, seat and lock an object. This almost always involves using passive(typically springs) and active(robot movement through sensor information)compliance in a cooperative manner. A GSFC engineer has pioneered numerous inventions in the area of passive compliance using a novel "Compliant Cable" [6] approach. With this approach cables can be wound and arranged in light weight, strong, and compact structures to provide vectored compliance. That is, the spring constant can be tailored independently in each of the six vectorial directions. The compliant cables, themselves, are composed of several strands wound around each other so they have a spring effect(coupled with impressive tensile strength); but with sufficient friction between the strands that the spring oscillations are damped out. This simple concept has been used as the basis for many devices in government and in industry of which only a portion would come under the heading of robotics. For example, these cables have proven to be excellent shock absorbers and vibration isolators for use in space and are also used extensively in the GSFC Robotics Lab. Commercial devices, based on this principle are already in extensive use. This is an important niche' market.

IV. COLLISION AVOIDANCE/MANAGEMENT SKIN.

Safety is a prime concern for robots operating in space; particularly when they are operating near humans and/or space structure. Thus, NASA is developing a collision avoidance/management skin wrapped around the robot arms. This skin will consist of an array of sensors each of which is called a "Capaciflector" [7](or capacitive reflector). Invented and developed at GSFC, this technique enables a capacitive sensor to be mounted in the immediate vicinity of the grounded robot arm and still "see" out to ranges an order of magnitude further than previously reported. For example, we routinely demonstrate picking up a human hand or a four in. dia. aluminium cylinder at ranges in excess of one foot using a 0.25 in. wide, four in. long strip of copper tape as a capacitive sensing element with an operating frequency of 20 khz and a potential of 10 volts. The previous state-of-the-art range of such a sensor is approximately one inch. Normally the electric field of a capacitive sensor couples both back into the ground plane and out towards an approaching object. The less the stand-off from the ground plane, the more the coupling into that ground plane and the less the coupling towards the approaching object. In the "Capaciflector" a shield is interjected between the ground plane and the sensor. This shield is driven at the same frequency and is in phase with the sensor. It is also at the same potential. However, it is electrically isolated from the tuning portion of the oscillator. Thus the sensor couples with an approaching object and changes the frequency of the oscillator (detection by standard fm techniques), whereas, the shield follows that oscillator frequency; but any coupling it may do to ground or the object does not effect operating frequency. The result is that for the sensor to couple to ground, its electric field must go around the shield to reach ground. Thus we

have the effects of a very large stand-off. The sensors and shield,however, may be part of a very thin flexible printed circuit board mounted directly on the ground plane. The commercial prospects for such a sensor are, of course, very significant. It will, no doubt, one day be a standard feature of many of the millions of industrial capacitive sensors at work throughout industry.

V. MAGNETOSTRICTIVE DIRECT DRIVE MOTORS.

Current robot motors are very high speed; but have weak torque compensated by using a transmission with extensive gearing. Since safety brakes are also required in these joints, these brakes must be located to act on the motor itself or the drive shaft on the motor side of the transmission to give them sufficient holding leverage. All these additions, compensations and restrictions lead to complications, lower reliability and controls problems. The magnetostrictive motor project addresses these concerns by developing a device that has outstanding torque density and is self-braking with the power off. This permits the power to be taken directly off the drive shaft, eliminating brakes and transmissions. The magnetostrictive phenomenon using the material Terfenol-D shows promise because it generates impressive forces (> 4 ksi) and has excellent frequency response (6 khz for 0.25 in. dia. rod). However, it also has two significant drawbacks, it has a very short stroke (0.001 in./in.) and low magnetic permeability (5) [8]. These present formidable engineering challenges.

Two engineering approaches are pursued (Fig. 3); type A using the classical "inch worm" approach and type B using an original (more promising) approach based on a roller locking technique. A proof-of-principle type A device has been successfully tested. It produced 9 ft-lbf stall torque (a record for an electric motor of this size), had a 800 microradian step size (outstanding for precision control), consumed 600 watts power and had a no-load speed of 0.5 rpm. Sound generated by the pounding of the clamping rods was surprisingly modest. The device was 10.25x4.50x4.25 in. and weighed 39 lbs [9]. The weight is not significant, nor is the low speed since no effort was made to control weight and limit the inertias of the moving parts at this stage of development. Also, small diameter rods were used so even this breadboard is fundamentally underpowered. Never-the-less it is clear that with development, this motor will become very competitive with torques on the order of 100 ft-lbf and no-load speeds near 20 rpm. We are now poised to begin work on prototype B, bringing it to the same level as A. It should exceed A in torque, speed and have outstanding efficiency.

From a commercialisation point of view, two more years of development will be required before these motors are ready. Ultimately, however, we are expecting a niche' motor which will significantly extend the state-of-the-art in applications requiring low speed, safe, high torque in a modest-sized package.

VI. ARTIFICIAL MUSCLE.

It is clear that mechanisms, and hence robotics systems, are limited because we do not have a linear motor/actuator which can perform the functions of the basic muscle. Such a motor must both perform at the level of the human muscle (strength, compactness, linear stroke, frequency response, and controllability) and be able to consume fuel and produce power independent of an umbilical or large battery for expended periods of time. We are not seeking to reproduce the human muscle; only its performance. This project is being initiated at a highly qualified university under NASA/GSFC direction. It is realistic and results and products are, we feel, a near certainty. However, they are approximately three years away. The eventual commercial implications of this work will, of course, be very significant.

SUMMARY AND CONCLUSIONS

Highlights of the NASA/GSFC program in advanced mechanisms and sensors have been presented. From this overview four things should be clear. 1. There is a tremendous amount of work to be done, from basic

screws to futuristic muscles. There are crippling inadequacies in our present capabilities every where. 2. The existence of these inadequacies is distorting and hindering the progress of robotics in general. 3. NASA/GSFC is doing everything humanly possible to plug some of the more glaring holes as quickly as possible. 4. There are important commercialisation opportunities to be realized throughout.

REFERENCES

1. Vranish, J.M. and Sharifi, Mohammad: The NASA/GSFC Split-Rail Parallel Gripper. Proc. 2nd European In-Orbit Operations Technology Symposium, Toulouse, France, September 12-14 , 1989.

2. Brown, J., NAVTROL Corp., Small Business Innovative Research Program-Controller demonstration at GSFC, July 27, 1990.

3. Vranish, J. M. : Rolling Friction Fingers. Research and Technology-GSFC Annual Report 1989, pp. 218-219.

4. Voellmer, G. Patent application entitled "Robotic Tool Change Mechanism", dtd. October 12, 1990.

5. Vranish, J. M. Invention Disclosure entitled "Spline Screw Fastening and End Effector System", dtd. October 22, 1990.

6. Kerley, J. Personal notes.

7. Vranish, J. M. and McConnell, R.L. Invention Disclosure entitled "Driven Shielding Capacitive Sensor ", dtd. August 20, 1990.

8. Terfenol-D notes, Publication of Edge Technologies, Inc. ETREMA Products Division Volume 3, #1, March 1990.

9. Glapa, S. Honey Robotics, Inc. Test Report dtd. October 10, 1990.

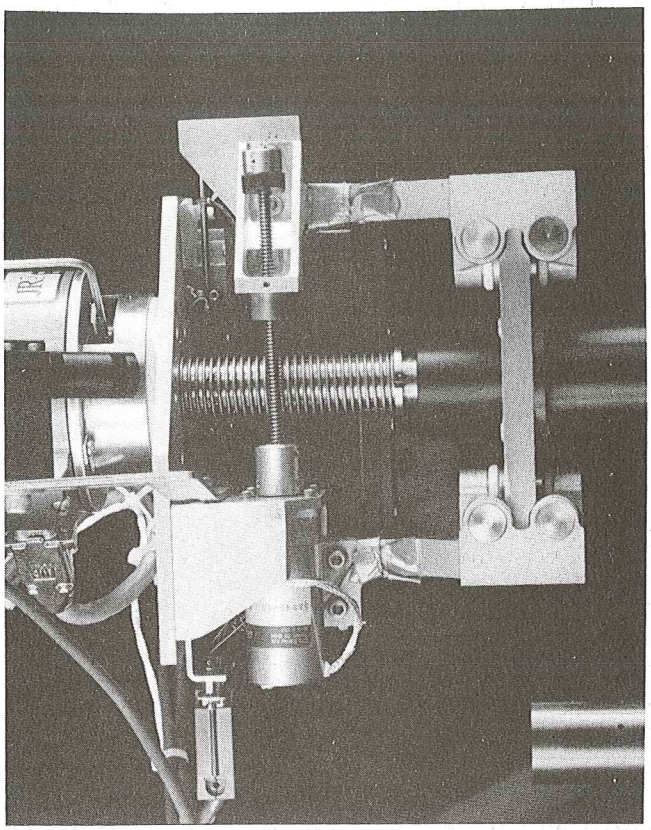


FIG. 1 GRIPPER NUT RUNNER

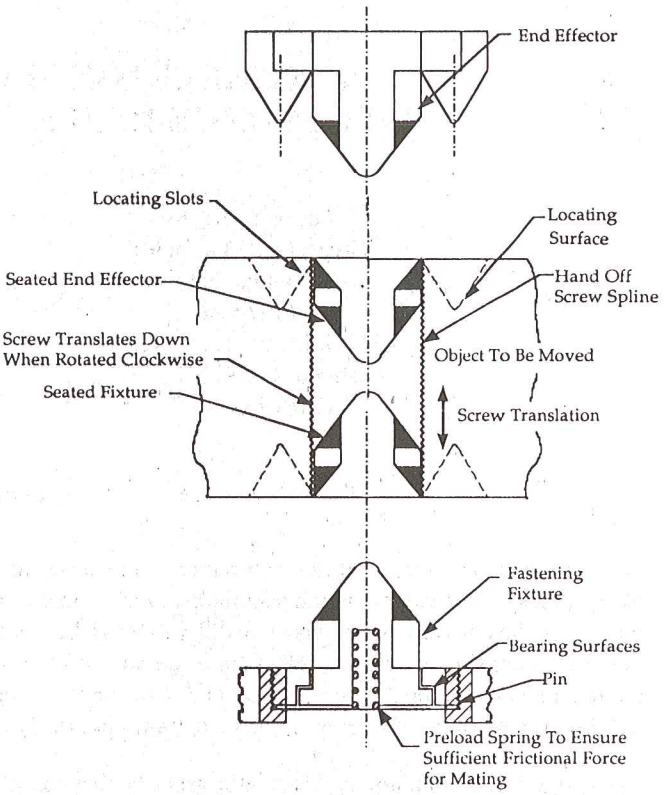


FIG. 2 FASTENING USING LOCKING SPLINES

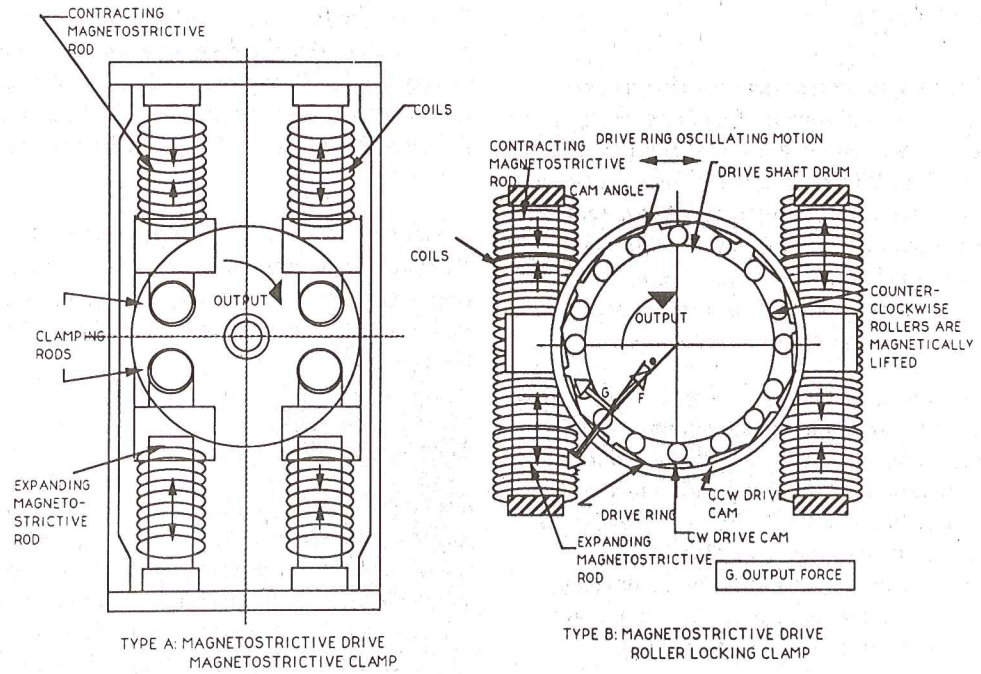


FIG. 3 MAGNETOSTRICTIVE DIRECT DRIVE MOTOR COMPETING DESIGN CONCEPTS

"EXOSKELETON MASTER CONTROLLER WITH FORCE-REFLECTING TELEPRESENCE"

James B. Burke
Stephen J. Bartholet
Odetics, Inc.
AIM Division
1515 S. Manchester Avenue
Anaheim, CA 92802-2907
(714) 758-0300

1st Lt. David K. Nelson
Det. 1, Armstrong Laboratory
Biodynamics and
Biocommunications Division
Wright-Patterson AFB, OH 45433-6573
(513) 255-3671

ABSTRACT

A thorough understanding of the requirements for successful master-slave robotic systems is becoming increasingly desirable. Such systems can aid in the accomplishment of tasks that are hazardous or inaccessible to humans. Although a history of use has proven master-slave systems to be viable, system requirements and the impact of specifications on the human factors side of system performance are not well known. In support of the next phase of teleoperation research being conducted at the Armstrong Research Laboratory, a force-reflecting, seven degree of freedom exoskeleton for master-slave teleoperation has been concepted, and is presently being developed.

The exoskeleton has a unique kinematic structure that complements the structure of the human arm. It provides a natural means for teleoperating a dexterous, possibly redundant robotic manipulator. It allows ease of use without operator fatigue and faithfully follows human arm and wrist motions. Reflected forces and moments are remotely transmitted to the operator hand grip using a cable transmission scheme. This paper presents the exoskeleton concept and development results to date. Conceptual design, hardware, algorithms, computer architecture, and software are covered.

INTRODUCTION

The purpose of a teleoperation system is to project an operator's manipulative and sensory functions into a remote environment, with the goal of performing work or extracting information. When using the system, an operator should feel that he has been projected into the remote environment. He should feel that he is at the work site, performing work with his own, unencumbered hands. Telepresent systems, through the use of sensors and manipulative capabilities, try to achieve this goal.

Perhaps the most important element in a teleoperation system is the interface between the human operator and the mechanical components of the system. This interface strongly influences both the system's performance the operator's perception of a remote environment. In the past, most master controllers have had six or less degrees of freedom and were kinematically similar to the devices they controlled. The human arm, however, is minimally represented by at least seven degrees of freedom.

Adequate feedback to the operator is essential for successful teleoperation. For example, the addition of stereo vision makes most remote, video teleoperated tasks significantly easier, if not possible. Force and/or audio feedback have been shown to be essential for tasks that involve a relatively high degree of physical interaction with a remote environment.

In support of research being performed at the Armstrong Research Laboratory on human factors issues in teleoperation, a better interface, in the form of a novel, seven degree of freedom, force reflecting master controller, is presently being developed.

Background

Teleoperation systems have historically allowed the accomplishment of tasks that were hazardous or inaccessible to humans. Master-slave systems have been used by the nuclear and other industries to remotely handle hazardous materials for over thirty years. The use of teleoperation technologies in submersibles has allowed man to explore and work in the deep sea, an environment that would oth-

erwise have been inaccessible. As such, the idea of a person projecting manipulative and sensory functions into a remote environment is not new.

The application of teleoperated system technologies in space has been a focus of research within the Nasa centers for much of the past decade. Teleoperation technologies, if sufficiently developed, could greatly facilitate satellite construction, servicing, and refueling. For defense applications, it could be advantageous to remotely operate in chemical, biological, or nuclear environments. In the commercial area, hazardous waste removal and other broadening markets demand advances in teleoperation technologies. With advances in computer, sensory, and manipulative technologies, the science of teleoperation is evolving.

Manipulative technologies have recently progressed beyond those currently used for teleoperation purposes. Dexterous, redundant degree of freedom manipulators are now commercially available. Dexterity implies that a manipulator has a relatively high payload capability. This quality is essential for the successful completion of many teleoperation tasks. Redundancy in the degrees of freedom (seven or more joints to control six degrees of freedom) allows one to influence the configuration of a manipulator in addition to controlling position and/or forces at the endpoint (in this paper, position refers to position and orientation and forces refers to forces and moments.) Configuration control can, among other things, be used to increase the dexterity of a manipulator or to avoid obstacles when working in a cluttered environment. Of the redundant manipulators that are currently under development, many are somewhat anthropomorphic in design, containing spherical joints at the shoulder and wrist, and a single revolute joint at the elbow. These manipulators, with high performance joint torque servos and anthropomorphic kinematics, are well suited for the application of kinesthetic master-slave teleoperation.

Progress in cable driven actuators has also pushed the state of technology that is available for teleoperation systems. An early example of a cable driven master is that of the Salisbury/JPL hand controller. Other, more recent examples are the Utah-MIT hand and WAM manipulator. Each digit of the hand is servoed to the position of an operator's finger joint through an antagonistic cable/tendon arrangement. The WAM manipulator demonstrates that cable transmissions can allow remotely mounted motors to apply relatively high torques at the joints of a compact, light weight manipulator structure. There are benefits that suggest the use of cable transmissions. Cable transmissions are relatively efficient, do not contain backlash and are relatively smooth in operation.

EXOSKELETON CONCEPT

The exoskeleton has seven degrees of freedom that anthropomorphically map those of the human arm. The shoulder roll, pitch, and yaw joints intersect at the approximate center of rotation of the operator's shoulder. The exoskeleton elbow pitch joint, although slightly offset from the human elbow for practical kinematic reasons, very closely and spatially follows rotations of the operator's elbow. The wrist roll, pitch, and yaw joints intersect at the approximate center of the operator's wrist. A conceptual rendering of the exoskeleton is presented in Figure 1.

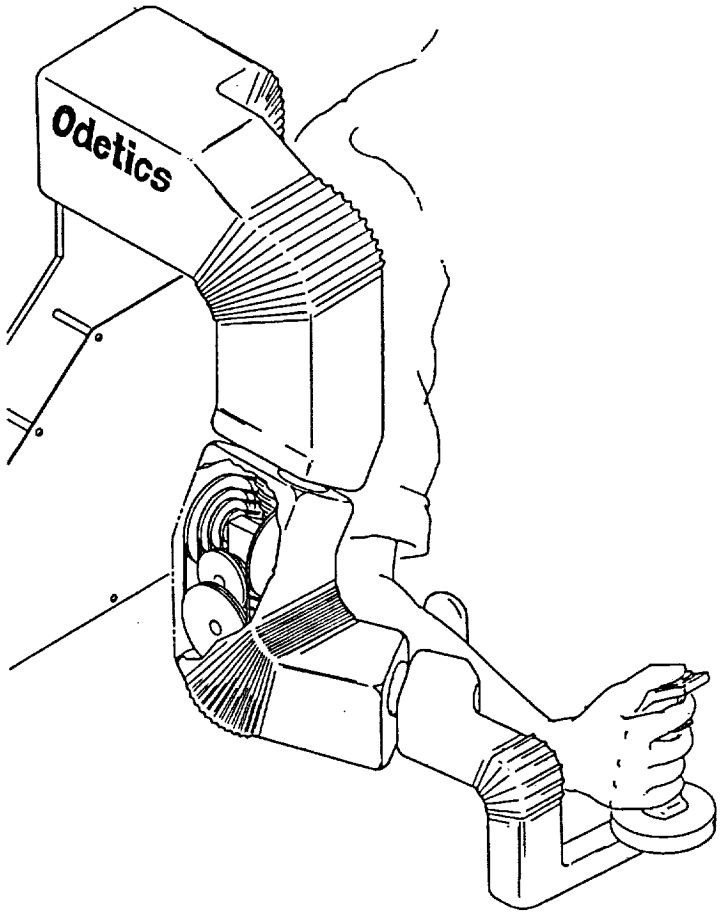


Figure 1 Exoskeleton Concept

Technical Description

Because the exoskeleton is designed to have a kinematic structure that complements the structure of the human arm, it provides a natural telerobotic interface. The system being developed has a number of unique characteristics:

- Because of the anthropomorphic similarity, the exoskeleton follows human arm motions. A single push plate is provided on the forearm. No straps or restraints are required.
- The exoskeleton is electrically actuated by remotely located motors. Torques are transferred to the joints through an antagonistic cable tendon transmission scheme. The use of cable transmissions will allow a very lightweight, low inertia master.
- Gravity forces on the master are compensated in software (at the joints). No additional mass or complex counterweighting scheme is required.
- Gravity compensation may be tuned by the operator. Additional compensation is available to cancel the operator's arm weight.
- The exoskeleton system exchanges data in base cartesian coordinates. The exoskeleton a generic master that may be coupled to any capable slave device.
- A force sensor, mounted at the exoskeleton grip, may provide data that can be used to servo out the effects of friction and/or compliance. Better force resolution will be available at the grip.
- A parameter that characterizes the exoskeleton arm configuration will also be generated. This parameter can be used to influence the configuration of a redundant slave arm.

The exoskeleton will provide a natural means for teleoperating a similar, possibly dexterous, redundant degree of freedom manipulator. It will allow ease of use without operator fatigue and will faithfully follow human arm and wrist motions.

BILATERAL ARCHITECTURE

The human operator is limited in his capacities to generate commands and perceive kinesthetic and proprioceptive sensory information in a bilateral loop. Component performance that surpasses the capacities of a human operator will not increase system performance. Telepresence thus provides a natural bound for master and slave performance requirements.

Relatively little research has been performed to directly determine the affects of sensory and command bandwidths on teleoperation system performance. One consensus is that the master, as a goal, should be capable of generating commands in the 5-10 Hz range. Although the human operator is able to perceive changes in vibrational frequency up to approximately 320 Hz, we are interested primarily in higher amplitude, kinesthetic force feedback. As a goal, the master should be capable of reflecting kinesthetic feedback in the 20-30 Hz range. Again, relatively little research has been performed in this area. These specifications are based primarily on the compiled opinions of scientists with previous teleoperation hardware experience [3]. An important point to note is that bandwidth requirements for the command and feedback sides of the loop are asymmetrical.

The physics of most bilateral systems are, in general, similarly asymmetrical. The slave usually has a relatively high inertia and the master has a relatively low inertia that is coupled to a stabilizing influence (the human operator). The master is thus able to meet higher bandwidth feedback requirements while the slave is generally capable of tracking only lower bandwidth commands.

Position-position, position-force, and force-force master-slave bilateral loops are theoretically feasible. Force-force loops, however, would not be practical for most situations.

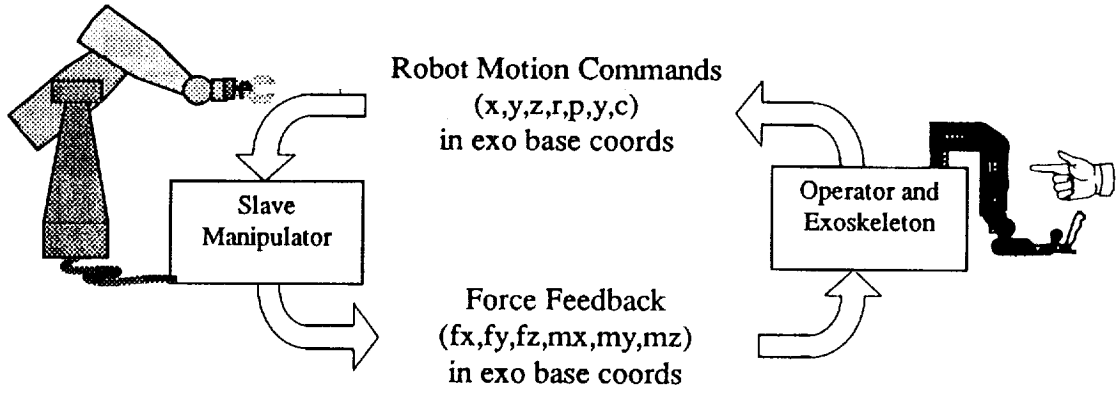


Figure 2 Position-Force Bilateral Architecture

When the master and slave are kinematically similar, position-position loops can be advantageous, as servos can be closed directly between the master and slave at each of the joints. In light of the asymmetrical bandwidth requirements for telepresence, however, a disadvantage is that feedback in such a system will be bandwidth limited by the slave inertia. Unless the bilateral slave has a relatively high position bandwidth, a position-force bilateral system (Figure 2) offers the greatest promise for telepresence.

Because we intend to create a generic master, one that could potentially be used with any capable slave manipulator, this consideration is deemed important.

Algorithms

Cartesian commands, based on the incremental position of the exoskeleton grip with respect to the exoskeleton base, are generated and sent to the slave. Cartesian forces, returned by the slave, are fed back to the exoskeleton operator by torque actuating the joints in a coordinated, controlled manner. A summary of these basic, open loop algorithms is presented in Figure 3.

Command Generation

Exoskeleton command generation algorithms consist of the forward kinematics and Euler angle transform algorithms. The forward kinematics routine solves the forward kinematics for the exoskeleton, generating a 4x4 matrix transform that relates locations in the exoskeleton grip coordinate frame to the base coordinate frame. The Euler angle transform routine extracts position and orientation commands in cartesian coordinates from the 4x4 matrix transformation.

Orientation angles in the exoskeleton command set are specified in terms of a Z-Y-X successive Euler angle representation.

The command set that is extracted by the Euler angle transform routine is absolute. However, commands that are sent to the bilateral slave are incremental, based on the difference between the absolute exoskeleton position and a set point that is selected when the grip deadman trigger is enabled (see REF for more information.)

The exoskeleton is a redundant degree of freedom device and, as such, more than one configuration may be possible for cartesian position set points in its workspace. A configuration command parameter is thus appended to the command set in order to pass information regarding the exoskeleton arm configuration to the slave.

If a redundant manipulator is coupled to the exoskeleton as a slave device, it should be possible to not only control the slave set point, but to influence the slave configuration. In teleoperation, the advantages of such a system would include natural master-slave obstacle avoidance and the ability to remotely configure the slave for maximum mechanical advantage.

Either an incremental wrist roll angle or an incremental elbow plane angle will be used to represent the exoskeleton configuration. The elbow plane angle is defined as the angle between a vertical plane passing through the exoskeleton shoulder apex and wrist apex and the plane formed by points at the shoulder apex, wrist apex, and a point at the center of the elbow joint.

Force Reflection

A recursive Newton-Euler formulation, similar to that suggested by Craig [4], is used to determine exoskeleton joint torques. At any instant in time, it is assumed that the

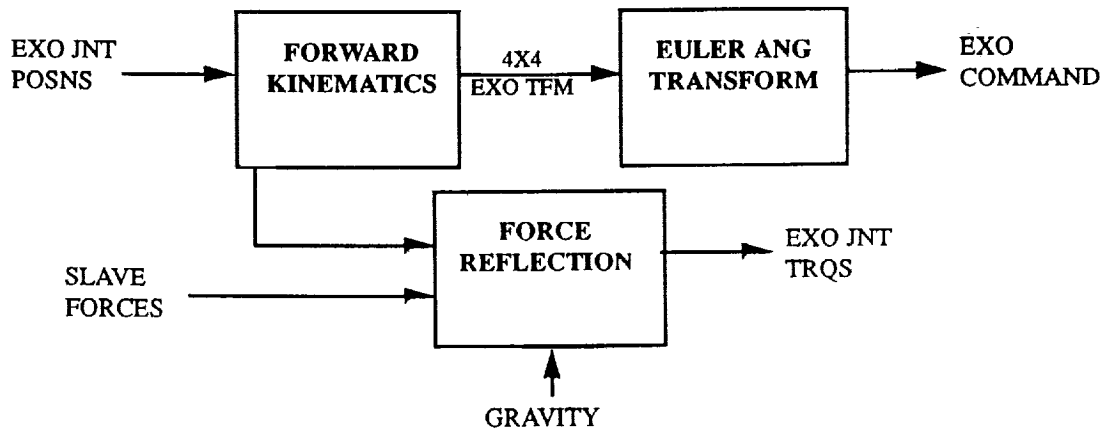


Figure 3 Exoskeleton Algorithms

exoskeleton joints are locked so that the arm becomes a structure. A force-moment balance for static equilibrium is then computed for each link in that link's coordinate frame. The inclusion of dynamic (velocity and/or acceleration dependent) forces in the calculation is not precluded.

Gravity forces on the arm are included by linearly accelerating the base frame of the exoskeleton upwards by a gravity acceleration constant. For the exoskeleton implementation, the gravity constant can be varied by the operator using a knob on the hardware user interface panel. Compensation may thus be tailored to the preferences of the operator or additional compensation can be added to support the operator's arm weight.

Algorithm Development Tools
Real-time software development for the forward kinematic and joint torque reflection expressions was inherently simplified by using the software tool Mathematica as a symbolic engine. Scripts have been developed to aid in the generation of real-time kinematic and dynamic models. Given a file of kinematic and mass parameters describing a system, the scripts generate real-time C subroutines.

Future Control Concepts

The implementation of more advanced control algorithms to alleviate the effects of friction and joint compliance may enhance future system telepresence. The arm is being developed so as not to preclude the implementation of such algorithms. Actuators and sensors for the arm have been chosen appropriately.

One concept that has been proposed would be to close a torque servo at each of the joints. Although closing the loops could prove to be a significant undertaking, this type of architecture would yield a very telepresent system and may be worth an investigation.

An additional control concept that has been proposed is that a six degree of freedom force sensor be placed at the grip of the exoskeleton. Conceptually, a force loop could be closed around the arm using this sensor. Friction in the joint transmissions effectively decreases the force resolution available at the exoskeleton grip. If the force at the grip were servoed to the commanded force, parasitic friction, compliance, and inertial forces would be negated. The exoskeleton system being developed will have provisions for the inclusion of a force sensor.

HARDWARE

Realization of the exoskeleton concept has provided significant engineering challenges. At the present time, engineering design is well underway and the procurement of mechanical and electrical system components has begun.

Mechanical Hardware

The exoskeleton arm and associated support structure, although integrally connected, were developed with somewhat different philosophies. Performance and cost were of primary importance in the design of the system. Where possible, arm mass and volume were minimized. Arm stiffness was maximized. Additionally, transmission stiffness, friction, and backlash were considered in the design of the arm. The support structure was also designed for performance, but mass, volume, and other packaging concerns were not deemed as important in light of system costs.

At the present time, link structural and transmission components, including shafts, bearings, pulleys, cables, and other associated hardware have been designed, and are presently being procured. Actuator hardware has been specified and is presently being procured. The exoskeleton support structure, including actuator mounting and electronic packaging has also been designed.

Exoskeleton Arm

The exoskeleton arm is the primary human interface to a teleoperation system. Design of the exoskeleton arm thus, arguably, has the greatest impact on telepresence as perceived by an operator.

Cable Transmissions
In order to transfer torques to the exoskeleton joints, a unique cable transmission scheme has been developed (see Figure 4). Antagonistic cable tendons, routed along the exoskeleton structure through banks of pulleys, transfer torques from remotely mounted motors to each of the joints. Remotely mounting the motors allows a minimal exoskeleton link structure. Additionally, a motor size limitation no longer precludes the active cancellation of gravity forces or the development of advanced control concepts for the arm. No counterweighting for the arm is required.

Early in the design effort, it was realized that double groove pulleys were required to allow joint rotations in excess of 90 degrees (cables are terminated at the driving and driven pulleys for positive power transmission.) In order to effectively transfer torques using a cable tendon transmission, the cables must be preloaded to half of their working load (to avoid hysteresis due to cable slack [5].) To satisfy cable loading requirements and, at the same

time, minimize bending friction, 7 by 19 multi-stranded 1/16th inch diameter wire rope was selected. Figure 4 presents a mock-up that was fabricated to verify the transmission concept. Because existing cable hardware could not satisfy volumetric packaging requirements for the arm, a custom termination scheme was also developed.

Exoskeleton Arm Joints

Another issue that was resolved relatively early in the design phase was how to efficiently mount pulleys on the exoskeleton structure. All shafts and non-driving pulleys rotate on bearings in order to minimize friction in the drive train. Conventional mounting schemes, however, do not satisfy packaging requirements for the concept (strict packaging requirements are evident in Figure 5.)

Previous experience at Odetics suggested that retaining compound could be used to fasten the bearings. Although the specification for Loctite R680 Aluminum-Aluminum bonds did not meet requirements for the arm, testing revealed that diametral clearance and surface finish significantly impact the strength of the bond. It was determined that shear load carrying requirements for the exoskeleton joints could be met if Aluminum surfaces were hard anodized and if design tolerances could be tightly held.

Exoskeleton links appear similar to those roughly shown in Figure 4, with the exception that some additional structure is required at the upper and lower arm twist joints. A photograph of the first two links, presently under assembly, is presented as Figure 6.

Joint Torque Actuators

Force reflection requirements have been selected to be 5 lb_f and 25 in-lb_f for each axis at the grip. Simulations, developed with conservative estimates for mass properties, transmission efficiencies, and actuator efficiencies, suggested that relatively large torques would be required for a successful system. Such torques would typically be obtained through the use of small motors coupled to sizable reductions. However, reflected actuator friction, linearly related to the reduction ratio, and reflected inertia, related to the square of the reduction ratio, will not be servoed out of the present, open-loop system. Small ratios are thus required to minimize reflected forces that would be perceived by an operator. Additionally, and perhaps more importantly, large reductions would limit actuator velocity and acceleration capacities, thus precluding the application of future servo concepts.

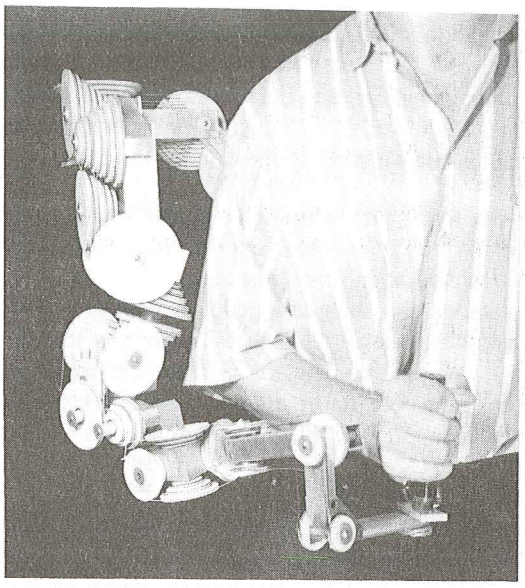


Figure 4 Transmission Mock-up

A number of actuator configurations were considered. Because the actuators are remotely mounted, packaging was not a priority in light of performance and cost requirements for the system. A relatively large, high performance brushless motor (Industrial Drives B206A) has been selected to meet system requirements. Low ratio, low backlash, low friction spur gear reducers (Bayside) are coupled to the actuators to obtain larger torques for the first four axes. The last three axes are directly driven.

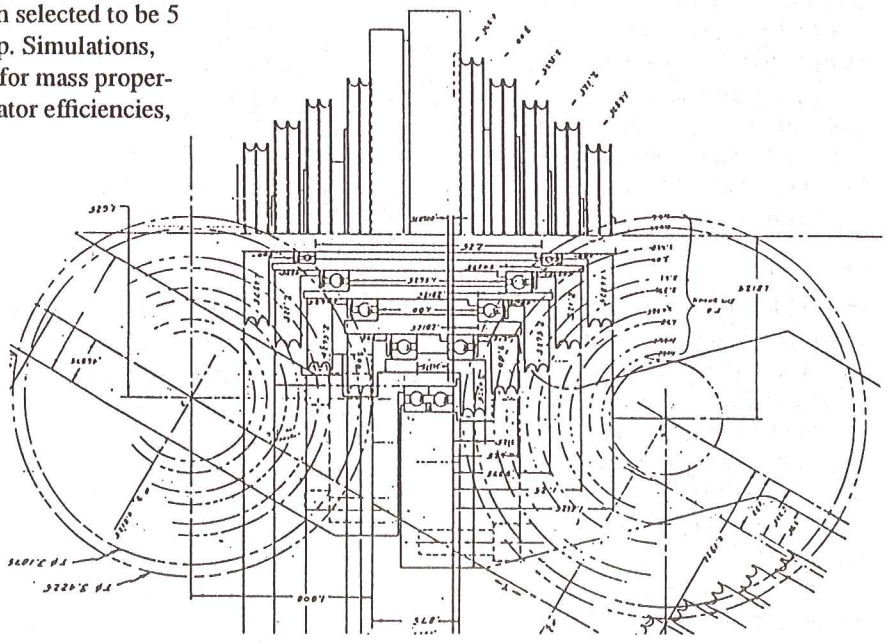


Figure 5 Shoulder Yaw Joint

Because the actuator hardware is commercially available, costs are not as prohibitive as the cost of a custom actuator. The motor has a high continuous torque capacity. Low armature friction and inertia allow the use of a gearhead without compromising system telepresence. A full sine-wave motor driver provides smooth, low speed DC torque. The motors that are being procured are modified to reduce cogging torque without compromising power density or linearity.

Because the gearheads use involute profile spur gears, high efficiencies can be realized. An additional benefit is that spur gear reductions are relatively linear and stiff, compared to harmonic or other reduction methods. This quality is very desirable from a control system perspective. We plan to use gearheads having minimal backlash so as not to preclude the future application of a torque loop or other compensation.

The present actuation scheme reflects a compromise between performance and cost. Although the development of custom actuators was considered, the benefits would have been primarily in packaging. Potential increases in system performance were deemed marginal in terms of the present goals and requirements for this program.

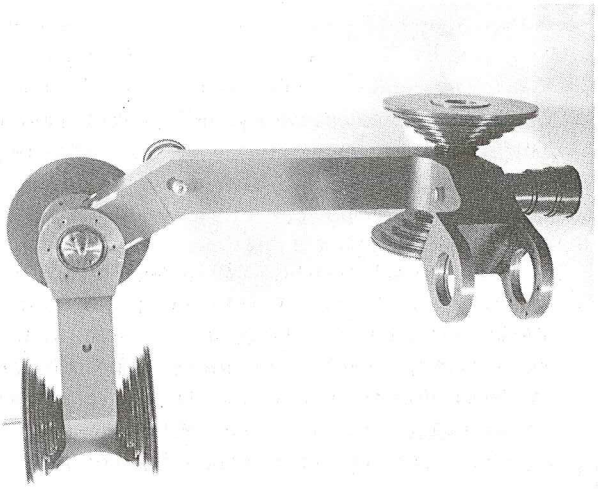


Figure 6 Shoulder Links

Support and Drive System

The exoskeleton support and drive system consists of a rigid aluminum structure that provides support for the arm, a mounting surface for the torque actuators, cable transmissions to couple the actuators to the exoskeleton shoulder pulley set, and a servo enclosure for the motor drivers, servo power supplies, and interface electronics. The support and drive system is presented as Figure 7.

The actuators are mounted to the rear, middle portion of the exoskeleton support structure. The position of each actuator can be adjusted in order to pretension the support structure pulley transmissions. Drive pulleys are located on the front (operator) side of the support structure and are covered by a clear plastic shield. At the lower portion of the support, an enclosed box houses the motor drivers, servo power supplies, and servo interface electronics. Space for a fan module is also provided. Removable panels have been designed into the sides of the enclosure to allow easy driver tuning or maintenance.

Although the support and drive system were also designed for performance, other packaging concerns (mass, volume, etc.) did not have as high a priority.

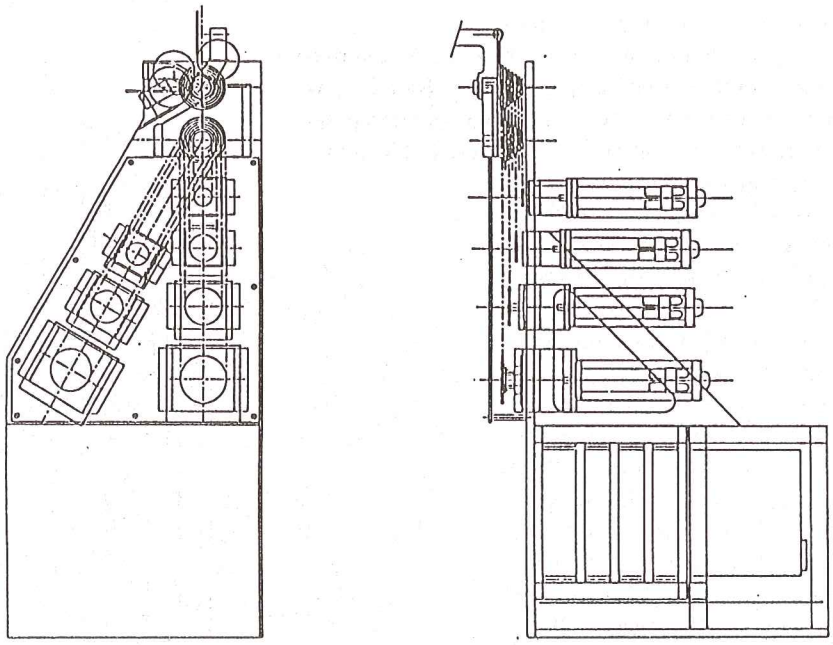


Figure 7 Support and Drive System

Electrical Hardware

Servo Electronics
The servo electronics consists of the motors, motor drivers, servo power supplies, servo power circuitry, and servo driver interface circuitry. Although components have been selected, the servo power interface circuitry has only been conceptually designed at this time. Detailed design of the interface circuitry is under way.

Computer / Interface Electronics
The computer and interface electronics consist of the processor enclosure and computer card cage, computer boards, instrumentation and computer power supply, the control panel, and the hardware interface electronics.

The exoskeleton controller enclosure contains 4 VME-based processor and I/O boards mounted in a 7 slot chassis. Three slots are thus available for future system expansion. The chassis has both J1 and J2 bus connectors in each slot. An integral fan cools the processor boards. Also housed in the exoskeleton processor enclosure are a 350 Watt computer and instrumentation power supply and custom signal conditioning and interface electronics.

The system hardware interface, consisting of the power switch, potentiometers for adjusting the command and force feedback gains, a potentiometer to adjust the gravity compensation gain, the emergency stop switch, a reset button, and status LEDs has been developed for the front panel of the processor enclosure.

In addition to the panel interface, two deadman switches are located on the exoskeleton grip. A Measurement Systems control grip has been specified to meet system requirements. The system has been designed so that emergency stop, motor temperature, and power supply error signals have authority over the Servo Enable deadman. Additionally, the Servo Enable deadman has authority over the Operate (bilateral communications) deadman. The Operate deadman output controls a relay that engages system communications. The states of both switches are available both at the hardware user interface and in software with other system states as semaphores.

In typical operation, the Servo Enable deadman will be depressed at all times to compensate for gravity. The Operate deadman will be used intermittently to initiate slave tracking of the master and to reflect slave forces.

Although servo enable and communications logic have been implemented in hardware to assure complete operator authority at all times, the real-time control loop also takes hardware states into consideration when generating position and joint torque commands. The gravity, command, and force feedback gains are scanned before being used in the real-time loop. there are thus no limitations on when the gains may be adjusted during normal use.

Computer System / Architecture

Exoskeleton computer hardware is VME-based. The VME (Versa-Multi-Europa) architecture offers an acceptable solution to a primary hardware/software problem in robotics. From a hardware standpoint, special purpose microprocessors (RISC based processors or DSP's) offer the best performance for inner control loop calculations. However, this hardware typically is not backed by mature software development tools. Additional and expensive software development time is usually required to write code for higher level tasks when these systems are used. For the higher level tasks, no additional performance is usually required. On the other hand, general purpose microprocessors provide good all around performance. Additionally, mature software development tools and a relatively large pool of experienced software developers exist for these processors. However, it is usually difficult to effectively implement both real-time and higher level code on a general purpose microprocessor and still meet real-time system demands. The VME bus architecture offers a solution: it allows several general purpose microprocessors to run in parallel. One or more processors can be dedicated to the real-time portion of the system, while others perform the higher level tasks.

A number of manufacturers offer off the shelf, high performance CPU and I/O boards for VME systems. This allows for flexibility in interfacing, expansion, and subsequent system modification. As advances in technology are made, higher performance SPU boards and higher density I/O boards can be integrated into the existing system relatively easily. Because a family of processors (Motorola 680x0) can be used, code is generally forward compatible and can be ported with minimal difficulty.

Computer and I/O Boards
A review of system requirements suggested that a single microprocessor would suffice for the present system. Figure 8 shows the configuration and hardware selected for the present system.

For development purposes, an Ethernet card has also been added to the VME system. This card allows a relatively transparent connection to an existing Sun workstation network. Because well documented, well used tools are available on the Sun network, the software development tasks are greatly simplified. Code can be designed, documented, compiled and tested in the workstation environment. Executable modules can then be downloaded to the vxWorks

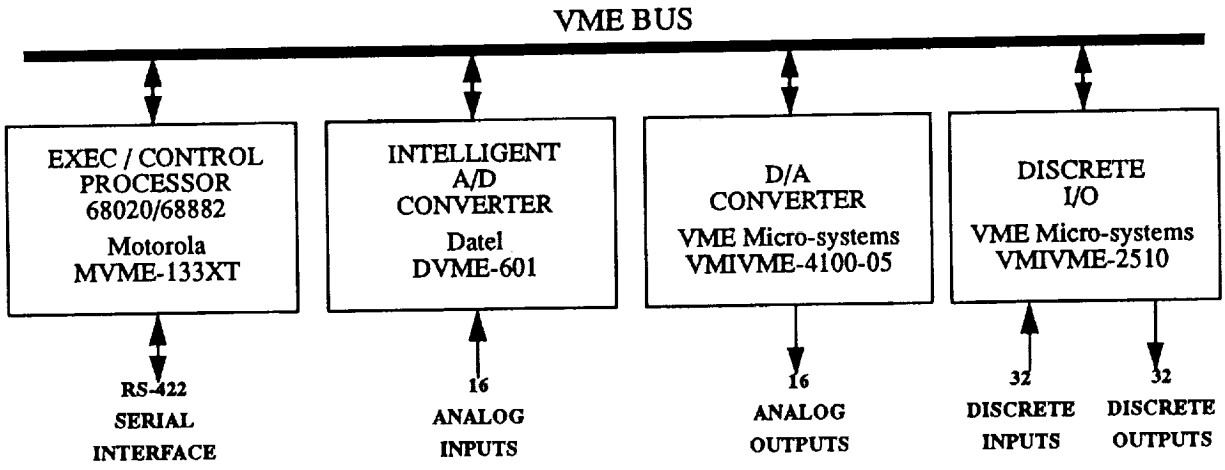


Figure 8 Computer Architecture / Hardware

real-time kernel on the host processor. The development interface, although not required for operation, provides the capability to adjust and display parameters while the system is in use.

Communications Interface

The communications interface between the master exoskeleton and robotic slave has been specified to be a high speed RS-422 serial interface. Thus, a single cable is all that is required to couple master and slave systems. The selected protocol calls for 192 kBaud, manchester encoded data. Data will be transferred in duplex between the systems. Routines are being developed to provide a buffered, interrupt-driven interface. The routines will provide a transparent software interface to the serial port. All data is transferred in scaled integral engineering units.

PROJECT RESULTS

Development is still underway and results are preliminary. Technical accomplishments that have been made to date include:

- Exoskeleton arm mechanical hardware has been fully designed. The transmission concept has been realized.
- Links 1 and 2 of the arm have been procured and assembly has commenced.
- The actuator concept has been developed and hardware has been specified. Parts are presently being procured.
- The support and drive system hardware has been designed and is presently being procured.
- The computer hardware architecture, software architecture, and communication protocols have been defined.
- Real-time computer system components have been specified, procured, configured, and assembled. Real-time software development is under way.
- All control algorithms required for basic bilateral teleoperation have been developed and ported to the real-time hardware.
- The hardware and software user interface protocols have been developed.

We are engineering a viable, working system based on the exoskeleton concept. At this time, we are confident that we will successfully meet program goals.

FUTURE POSSIBILITIES

The advantages and unique kinematic configuration of the exoskeleton suggest a number of interesting system possibilities.

- The kinematic redundancy in the exoskeleton is very similar to that in a spherical-revolute-spherical seven degree of freedom robotic arm. The exoskeleton may thus be used to naturally influence the configuration of a redundant slave arm based on the configuration of the operator's arm.
- Because the exoskeleton joints anthropomorphically map those of the operator, individual joint feedback (configuration force feedback) should be possible. This concept could be used for teleoperation with obstacle avoidance assistance.
- Right and left versions of the exoskeleton could be used together to allow natural, dual arm, coordinated control of teleoperated slave manipulators.

REFERENCES

[1] "An Exoskeleton Master Arm, Wrist, and End Effector Controller with Force Reflecting Telepresence," U. S. Department of Defense SBIR Phase I Final Report, Contract No. F33615-89-C-0587, May 1, 1989.

[2] "An Exoskeleton Master Arm, Wrist, and End Effector Controller with Force Reflecting Telepresence," U. S. Department of Defense SBIR Phase II Proposal, Contract No. F33615-89-C-0587, September 15, 1989.

[3] Brooks, Thurston L. "Teleoperator System Response for Nuclear Telepresence" *Proc. ANS 5th Topical Meeting on Robotics and Remote Systems*, Albuquerque, NM, February 25-27, 1991.

[4] Craig, John J. Introduction to Robotics: Mechanics and Control. Reading Massachusetts: Addison-Wesley Publishing Co., 1986.

[5] Townsend, William L. "The Effect of Transmission Design on Force-Controlled Manipulator Performance," Cambridge, Massachusetts: Ph.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1988.

REMOTE SYSTEMS DEVELOPMENT

by
R. Olsen, O. Schaefer, and J. Hussey
Space and Electronics Division
Grumman Aerospace Corporation
Bethpage, New York 11714

ABSTRACT

Potential space missions of the nineties and the next century require that we look at the broad category of remote systems as an important means to achieve cost-effective operations, exploration and colonization objectives. This paper addresses such missions, which can use remote systems technology as the basis for identifying required capabilities which must be provided. The relationship of the space-based tasks to similar tasks required for terrestrial applications is discussed. The development status of the required technology is assessed and major issues which must be addressed to meet future requirements are identified. This includes the proper mix of humans and machines, from pure teleoperation to full autonomy; the degree of worksite compatibility for a robotic system; and the required design parameters, such as degrees-of-freedom. Methods for resolution are discussed including analysis, graphical simulation and the use of laboratory test beds. Grumman experience in the application of these techniques to a variety of design issues are presented utilizing the Telerobotics Development Laboratory which includes a 17-DOF robot system, a variety of sensing elements, Deneb/IRIS graphics workstations and control stations. The use of task/worksite mockups, remote system development test beds and graphical analysis are discussed with examples of typical results such as estimates of task times, task feasibility and resulting recommendations for design changes. The relationship of this experience and lessons-learned to future development of remote systems is also discussed.

INTRODUCTION

The inherent capacity of humans reaches its full potential when we learn to extend our reach beyond our immediate environment. Whether it be by humans traveling into space or by sending our intellect and physical capacities via robotic vehicles, while we remain behind, our species must always explore and develop the next frontier. These philosophical reasons for human involvement in space encourage us to develop ways to extend our reach while maintaining safety, practicality, and the use of resources within acceptable bounds. It is for these reasons that *remote systems*, i.e. systems that operate with a degree of self-contained intelligence to perform useful functions for humans but at a location removed from the presence of humans, have been a "growth" area since the beginning of the space program. Within the framework of this definition of remote systems, the human can provide varying degrees of control of the remote elements - ranging from simple changes of commands, e.g. between two positions of an antenna gimbal drive, to highly interactive control of a robot manipulator which sends measured force information to a remote human operator, to a fully autonomous robot which carries out a complete task or even a complete mission at the request of a human. Remote systems, in this context, apply to a wide range of applications (Fig. 1) from low earth orbit (LEO) through geostationary earth orbit (GEO) to lunar and planetary missions, and include "robots" which perform manipulation functions, and "remote vehicles" which provide translation capability.

This paper is concerned with how these systems are developed through a process of mission analysis, identification of design issues, and the use of various available techniques for their resolution. It also illustrates how the remote tasks proposed for future space missions correspond closely to a myriad of potential applications on the earth (Fig. 2).

Grumman has been involved since the 1970s with the development of remote systems beginning in an "inner space" application with robot arms on the Ben Franklin submersible and remote maintenance devices for the nuclear industry to involvement in the 1980s with space systems for satellite servicing, remote military ground vehicles and flexible assembly systems for aircraft manufacture. The emphasis has generally been on a wide variety of tasks in an unstructured environment.

REMOTE SYSTEMS

A remote system as defined above does not begin to have real meaning until the definition is expanded in the form of the general system architecture presented in Fig. 3. This figure presents the concept of a human, the user, in one location or environment controlling effectors and tools which are the means of performing desired functions or tasks, in some other location or environment and utilizes sensors for task control and user feedback. The sensors provide the spatial/temporal information of the gaming area for remote control of mechanisms and the remote sensory perceptions of the user. The degree of perception of task conduct and completion is a key to the strategy of using remote mechanisms intelligently.

The user receives sensory information and enters commands through a user interface while information is controlled and manipulated between the two environments by processing elements. The processing, as illustrated, is gen-

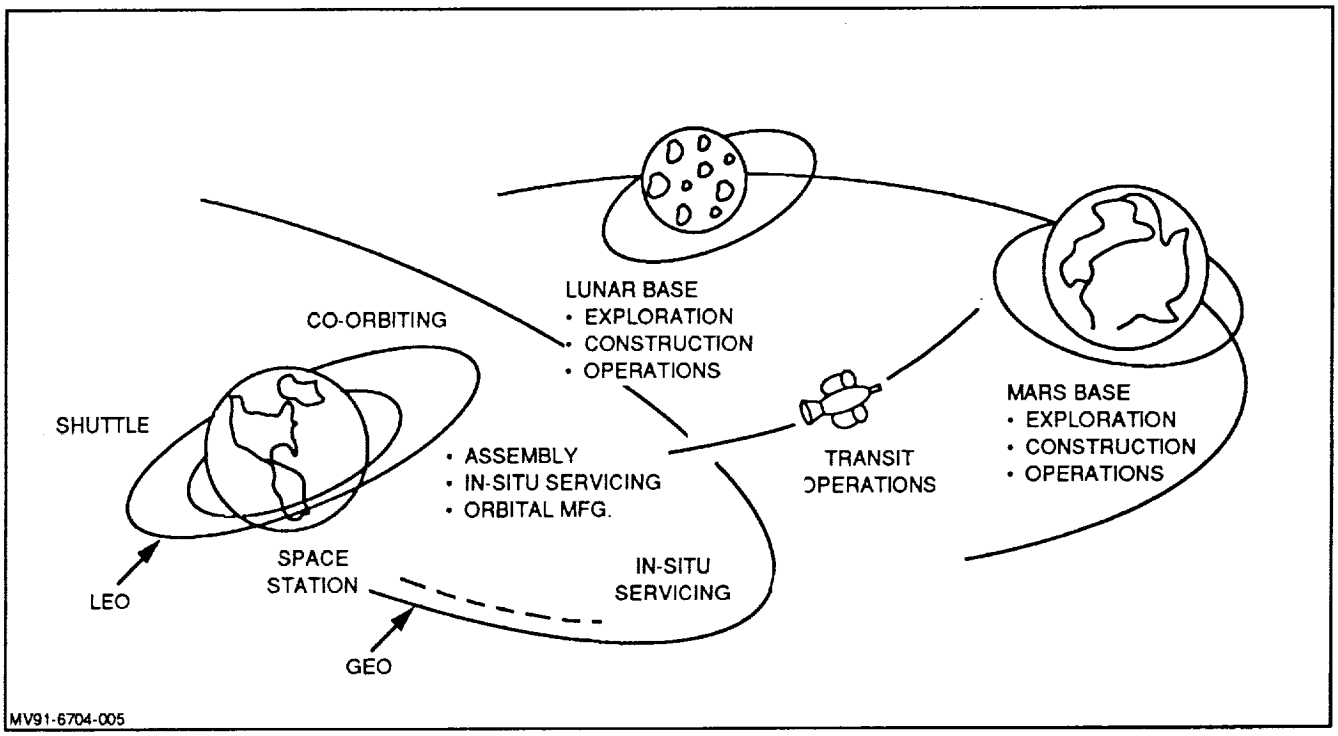


Fig. 1 Future Remote Systems Scenarios

erally divided between the two environments, with the processing complexity in either location being a function of the quantity of information and the level of sophistication of the computations required. When most of the processing is at the user location, the control approach is described as teleoperation. When almost all of the processing takes place at the remote site, the system is described as autonomous. In the latter case, the user is mostly a periodic observer of the task being performed with the ability to redirect or take charge at any time.

The communications element between the processing in the two environments, which is shown in the figure, provides for the passage of signals and commands. In some cases, this element may be nothing more than a cable

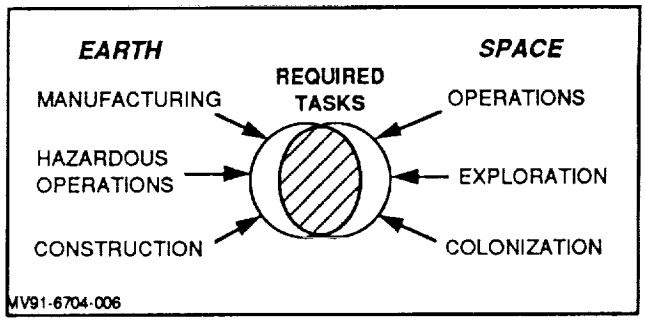


Fig. 2 Similarity Between Earth & Space Remote Systems Tasks

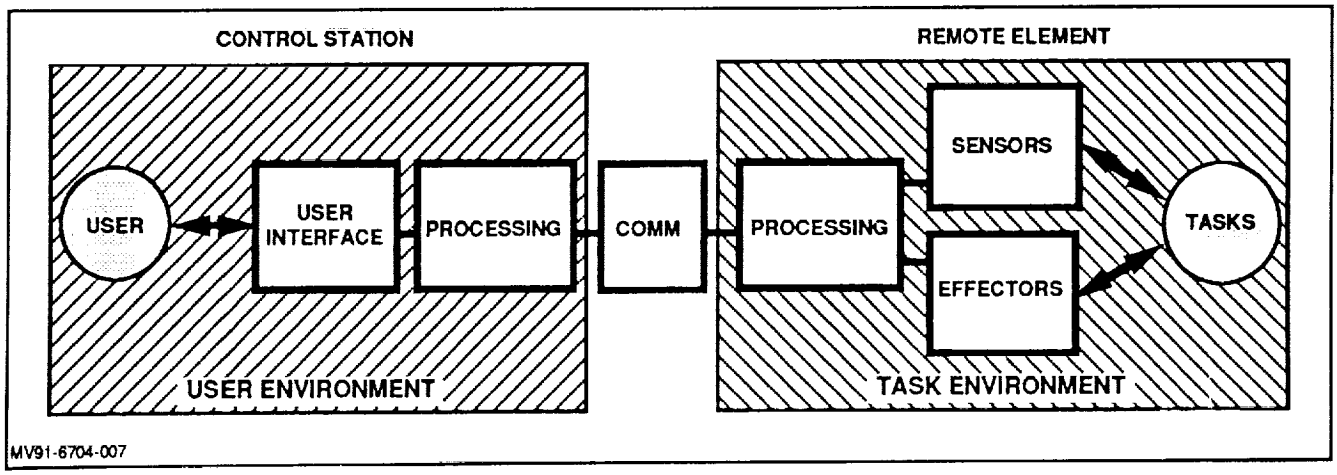


Fig. 3 Remote System General Architecture

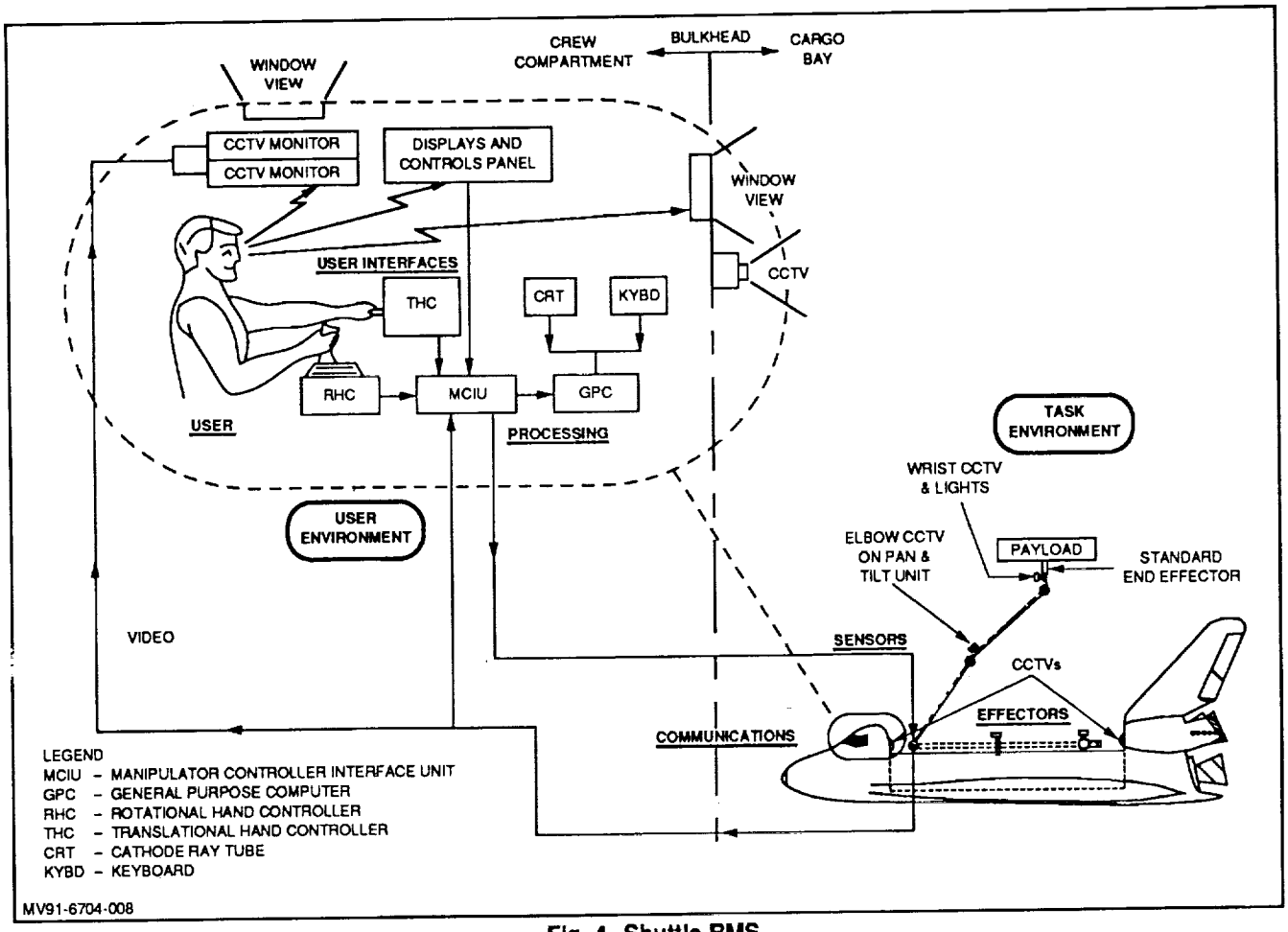


Fig. 4 Shuttle RMS

between the two environments. It becomes especially important when long distances are envolved such that wireless communication is required. Often the communications link is characterized by a long time delay which may be a significant factor in developing a remote system.

Figure 4 illustrates the above architecture applied to a specific remote system, the shuttle's Remote Manipulator System (RMS), with the system elements identified.

FUTURE MISSIONS

The development of remote systems for space is very dependent on the wide range of anticipated future missions (Fig. 5). These missions can generally be grouped into three categories: operations, exploration and colonization. The operations missions have already begun as exemplified by the use of the shuttle RMS for a variety of deployment and servicing support functions in more than 20 missions. The 1990s will see an increase in operations with shuttle and space station tasks carried out by U. S. and foreign remote systems such as the Canadian Special Purpose Dexterous Manipulator and the Japanese Small Fine Arm. It is difficult to project too far into the future but it appears that in addition to increasing operations functions, the exploration and, eventually, the colonization functions will rely on remote systems to achieve objectives in a cost-effective manner.

Potential applications of remote systems for these missions are presented in Fig. 6 which also identifies some of the key technologies required to provide the capability. There is also a group of related applications of remote systems required on earth (Fig. 7). Much of the technology required is generally the same although specific differences may exist because of unique environmental or functional requirements. Nevertheless, it is believed that the development of remote systems for earth and space applications should take place in concert to a large extent. The methodology for such development, presented in subsequent sections for space remote systems, is applicable to terrestrial systems and, in fact, benefits can be obtained through joint efforts in selected areas.

REQUIREMENTS/ISSUES

The development of remote systems (Fig. 8) starts with the decomposition of the mission requirements into goals, tasks and subtasks, and the characteristics of the mechanism and user/work environments. The functional decomposition lends itself to development of a relational data base that will maintain traceability through the design concept development stage. This initial stage of decomposition must characterize the performance indexes, constraints, time functions, data items, and components. To achieve this de-

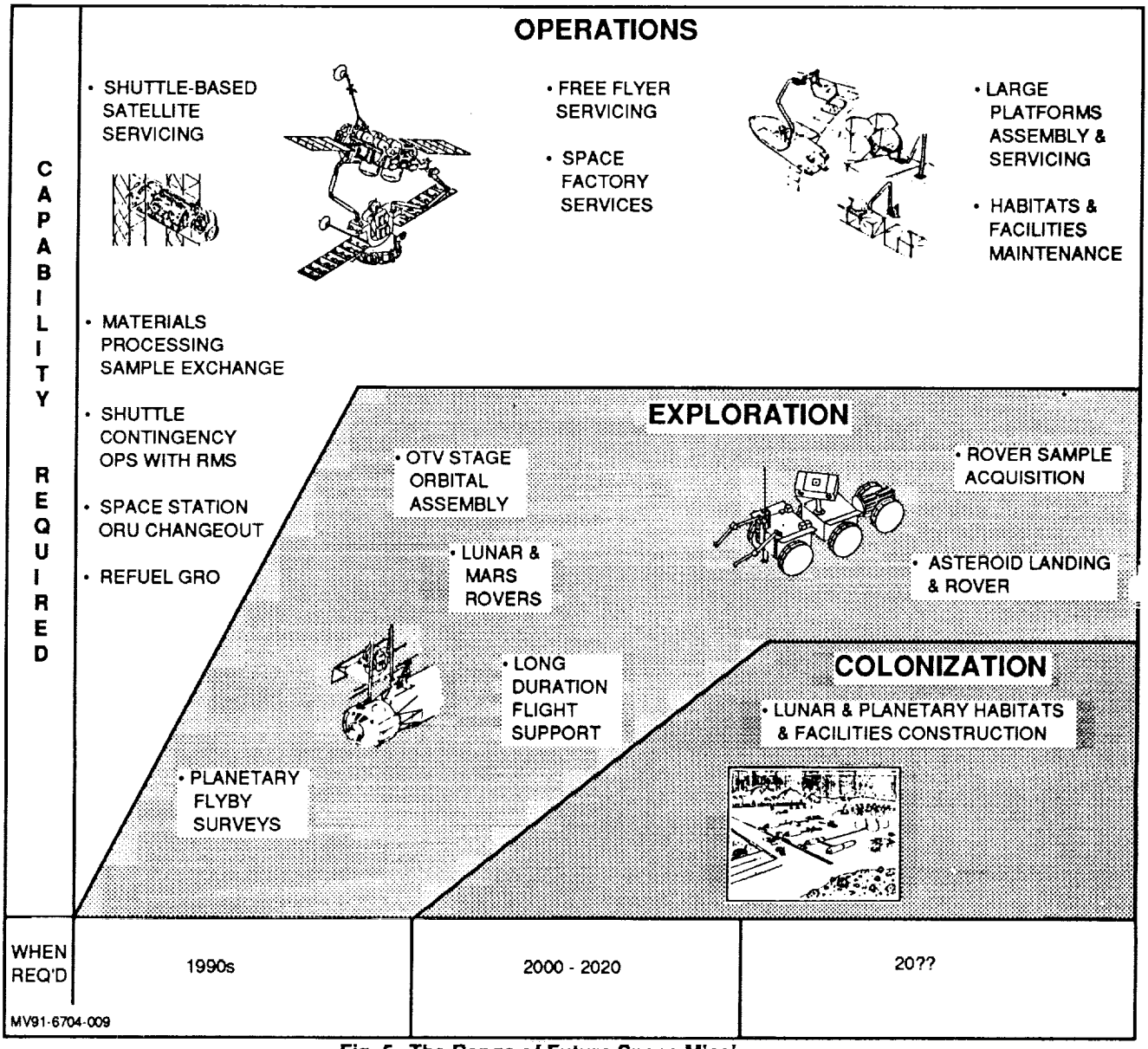


Fig. 5 The Range of Future Space Missions

composition, the mission definition must therefore characterize the payload size, operational volume, physical constraints, task environment characteristics, and any special functions/characteristics that are germaine to the proposed mission.

The objective is to develop a set of requirements that define the boundaries within which the remote system must function to accomplish the mission tasks.

From this base a series of trades can be conducted to establish the global or architectural issues of the design. Although the structured environment of the factory is ideal, the generally more unstructured environment of space systems can be viewed as similar, but with more interactions required to address the uncertainties (i.e. constraints).

The development of system requirements requires careful consideration of the fundamental design issues relative to the mission-related factors which establish remote system requirements. Figure 9 indicates which design factors for remote systems are affected by five mission factors which generally cover all system requirements. Payload size includes not only the volume but unusual shapes and configurations. Operational volume is the volume at the task site which must be reachable physically by the manipulator arms and visually by the sensors. At the control station, the volume requirement impacts the design of controllers which require operator movement.

Physical constraints refer to all types of mission factors which may limit or prevent some performance capabilities of the system design. An example relative to the manipulator arms would be task site characteristics which require reaching around obstacles to perform a task. Relative to the

control station, a physical constraint would be an existing display panel design the prevents the easy incorporation of some display features especially suitable for telerobotic operations.

Task environment factors include items such as natural lighting conditions at the task site which may have a major impact on sensor performance requirements. Another task environment example is physical separation between the robot and control station which results in significant transmission time delay.

The special functions category is meant to cover unique mission requirements which do not fit into the previous categories but which may have a significant effect on system design. Examples are missions requiring handling of cryogenic equipment and operations with hazardous fluids.

There are many issues associated with remote systems design which emanate from the above system requirements and which generally must be resolved before a design can be finalized. Major issue areas (Fig. 10) are the mix of humans and machines, the particular remote system's characteristics, the compatibility of the worksite with the remote system elements, and the required workstation features. Each of these areas is discussed below in terms of specific items which may require resolution.

The *mix of humans and machines* is of special interest because it involves reaching a balance between human capabilities and the limitations of technology relative to the human intellect. The level of autonomy/processing of the remote system must be sufficient to meet the mission requirements and is dependent on the specific tasks and the range of variations in the task environment. High levels of autonomy require that the capabilities of the remote systems technology be sufficiently developed to properly characterize the dynamic task environment from sensor information, to compute the necessary response to carry out the mission, and to provide the means for creating this response. The level of feedback/communications to the human from the remote system is another important element because it significantly impacts the communications link requirements. The decision level of the human in the performance of the remote tasks is another factor which can vary from a high level that only decides on the next task to be performed based on successful completion of a task, to low level decision making such as the path planning choices made during a task.

An important element determining the mix of humans and machines concerns the evaluation of hazards which can affect mission success, especially relative to systems where humans are present. The mix will be different if the tasks involve operations which can threaten humans if not performed properly or under failure conditions.

The *remote system characteristics* area basically concerns the design parameters for the remote system relative to the particular functions to be performed. It is primarily described by the physical size and capacity of the effectors, degrees of freedom of mechanisms such as manipulators, end effectors/tools for interfacing with the tasksite, and vision/lighting capabilities. The goal is a remote system which provides the needed performance levels without excessive capacity.

USER GROUP	APPLICATIONS	TECHNOLOGY REQUIREMENT
PERIODIC ORBITAL OPERATIONS	• S/C SERVICING AND MAINT. • S/C ASSEMBLY IN ORBIT • CONTINGENCY OPERATIONS	• TELEOPERATED SYSTEMS • BILATERAL FORCE-REFLECTING, DEXTEROUS 2-ARM MANIPULATOR • VISION/LIGHT SENSORS
PERMANENT SPACE OPERATIONS	• MATERIALS PROCESSING OPERATIONS • HABITAT MAINTENANCE • PAYLOAD ADJUSTMENTS & RECONFIGURATION	• AUTONOMOUS SERVICING SYSTEMS • PRECISION MANIPULATORS • ADVANCED SENSORS
EXPLORATION MISSIONS	• PLANET MAPPING & INTERPRETATION • ROVER VEHICLE PLANETARY EXPLORATION • EARLY SURFACE CONSTRUCTION TECHNIQUES	• ADVANCED TELEOPERATED SYSTEMS • HIGH RESOLUTION TV TRANSMISSION • HIGH RELIABILITY SYSTEMS • MOBILITY SYSTEMS
COLONIZATION MISSIONS MV91-6704-010	• SURFACE MINING • CONSTRUCTION • TRANSPORTATION VEHICLES	• AUTONOMOUS SERVICING SYSTEMS • PRECISION MANIPULATORS • ADVANCED SENSORS

Fig. 6 Remote Systems Applications – Space

The *tasksite compatibility* area concerns the characteristics of the task environment which have a direct bearing on the remote system design. It includes the nature of the terrain for mobile remote systems. Cooperative features refer to physical characteristics which simplify the design of the remote system for performing the required functions such as easily accessible attachment fittings at all tasksites.

The variations and characteristics of work articles, i.e. items to be handled in some way by the remote system, are very important measures of tasksite compatibility. Safety of the remote system is a function of the compatibility of the tasksite in terms of avoiding tasksite features which may inadvertantly damage the remote system during its operation. Remote systems working in proximity to humans is a special case which must be addressed.

USER GROUP	APPLICATION	TECHNOLOGY REQUIREMENT
SURGERY & REHABILITATION UNITS	• ISOLATION WORK OPERATIONS • MICROSURGERY • AUTOMATED SURGERY IN EMERGENCY • HAZARDOUS MATERIAL HANDLING • PATIENT REHABILITATION SERVICES	• ADVANCED TELEOPERATED SYSTEMS • BILATERAL, FORCE-REFLECTING, DEXTEROUS TWO-ARM MANIPULATOR • HIGH RESOLUTION VIDEO
NUCLEAR POWER	• PLUTONIUM MANUFACTURING • SPENT FUEL REPROCESSING • STEAM GENERATOR INSPECTION • DECONTAMINATION • ASSEMBLY, DISASSEMBLY, REPAIR	• TELEOPERATED SYSTEMS • BILATERIAL FORCE REFLECTION • HIGH RESOLUTION TV TRANSMISSION • VACUUM COMPATIBLE MANIPULATOR
ELECTRONICS	• HIGH-SPEED ASSEMBLY OF "BOTTLENECK" PARTS • CLEAN ROOM OPERATIONS	• TELEOPERATED SYSTEMS • ADVANCED TRANSDUCERS & SERVO CONTROLS • VISION SENSORS FOR REGISTRATION
OIL EXPLORATION	• OFF-SHORE OIL RIG MAINTENANCE • WELL HEAD MAINTENANCE • INSPECTION	• TELEOPERATED BILATERAL FORCE REFLECTING SYSTEM FOR HIGH-PRESSURE, CORROSIVE ENVIRONMENT • REMOTE SENSORS/VISION SYSTEMS
MINING	• UNDERGROUND OPERATIONS • HIGH-WALL SURFACE MINING • OPERATIONS • THIN SEAM MINING OPERATIONS • DRILL BIT AND BOLT INSTALLATION	• TELEOPERATED SYSTEM • DEXTEROUS HEAVY DUTY MANIPULATORS • ROBUST TV TRANSMITTING SYSTEM • ADVANCED LIGHTING SYSTEMS
CHEMICALS	• TOXIC MATERIAL HANDLING • LABORATORY OPERATIONS	• TELEOPERATED SYSTEMS • BILATERAL FORCE REFLECTION • HIGH RESOLUTION TV AND VISION PROCESSING
CONSTRUCTION	• BRIDGE MAINTENANCE • STRUCTURAL ASSEMBLY OPERATION • UNDERWATER STRUCTURAL ASSEMBLY • HIGH RISE BUILDING & BRIDGE ASSEMBLY	• TELEOPERATED BILATERIAL FORCE REFLECTION SYSTEM • HEAVY DUTY, HIGH STRENGTH 2-ARM MANIPULATOR • ROBUST VISION SYSTEMS
FIRE FIGHTING & EMERGENCY UNITS	• OPERATING IN RESTRICTED HOSTILE ENVIRONMENT • MUNITIONS REMOVAL	• LOW COST, RELIABLE, TELEOPERATED BILATERIAL FORCE REFLECTION SYSTEMS • LOW COST MULTISPECTRUM VISION SYSTEM

MV91-6704-011

Fig. 7 Remote Systems Applications – Earth

The area of *control stations* essentially covers all the issues associated with the design of the remote system at the location of the human user. It includes the types, quantities and configuration of controllers, the types, quantities and arrangement of displays, and the human factors aspects of the control station design.

SYSTEMS TECHNOLOGY

Remote system concept designs are very heavily influenced by the cost and availability of technology. A summary of required technology availability is shown in Fig. 11. To ease the introduction of new technology, remote systems should utilize open architectures with robust interfaces, for example, the NASA/NBS Standard Reference Model (NASREM) software architecture. Such a functional architecture imposes standard requirements on module interfacing, synchronization, communications and global memory access to support a hierarchal development, by levels, from Task Planner, to Path Planner to Trajectory Planner to Servo Control Level. It is designed to incorporate all levels of remote system automation. The standard interfaces provide the software hooks necessary to incrementally upgrade remote systems as new capabilities develop in processing, sensors, effector mechanisms and autonomous systems.

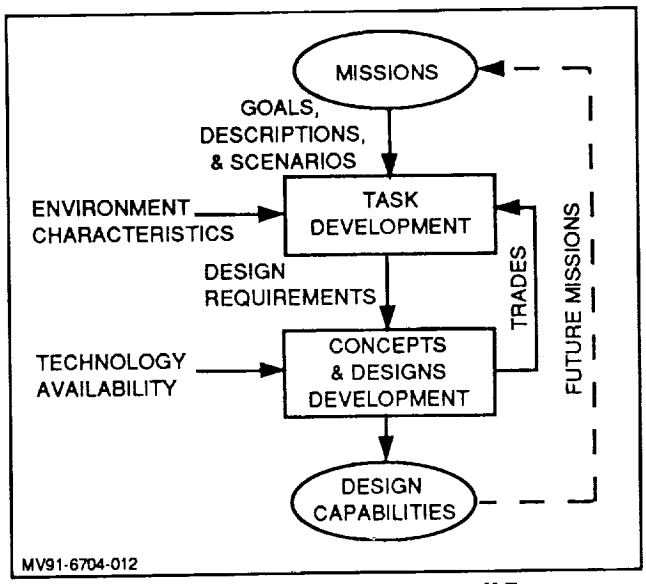


Fig. 8 Remote Systems Design Tradeoff Process

In the hardware area, the use of standards provides the guidelines for minimum levels of performance and functionality and tends to improve availability and reduce costs. It is also the driving force behind the third-party vendor products so heavily relied on. The present state-of-the-art in processors is being driven by the tri-service Joint Integrated Avionics Working Group (JIAWG) selections of 32 bit Instruction Set Architecture (ISA) standards for Reduced Instruction Set Computers (RISC) namely the MIPS R3000 and Intel i960 VLSI chips that produce 50 MIPS and the Parallel Intermodule Bus (PI Bus) wide bandwidth data bus specifications. These specifications are particularly meaningful since high-speed, low-power and wide bandwidth computations capabilities are the core technologies necessary to solve the remote system technology problems.

Another area requiring increased advanced development is the field of computation algorithms, that are targeted at the low-level parallelism available in matrix/vector processing architecture, for inverse kinematics/dynamics, sensor signal processing and sensor fusing. At present an order of magnitude speedup over state-of-the-art systems is needed. The tasks of work environment feature extraction and identification are presently mired in visible bandwidth vision system processing. The ultimate solution probably lies in a sensor fusion approach that requires a multi-spectrum sensor and/or a near field range/range rate sensor to augment the diffraction/disperson problems inherent in feature sensors. These technology areas are a key to remote system motion planning and execution.

Last but not least is the human interface. All too often we attempt to solve our technology problems by "letting the operator do it" without full knowledge of the cognitive work load/overload we create. The human problems of motion and depth perception, detection and recognitiion, and general task knowledge when added to the environment problems of communications delays, sensor perception, lighting and work area uncertainties have created a human factors nightmare. Therefore, there must be a concerted develop-

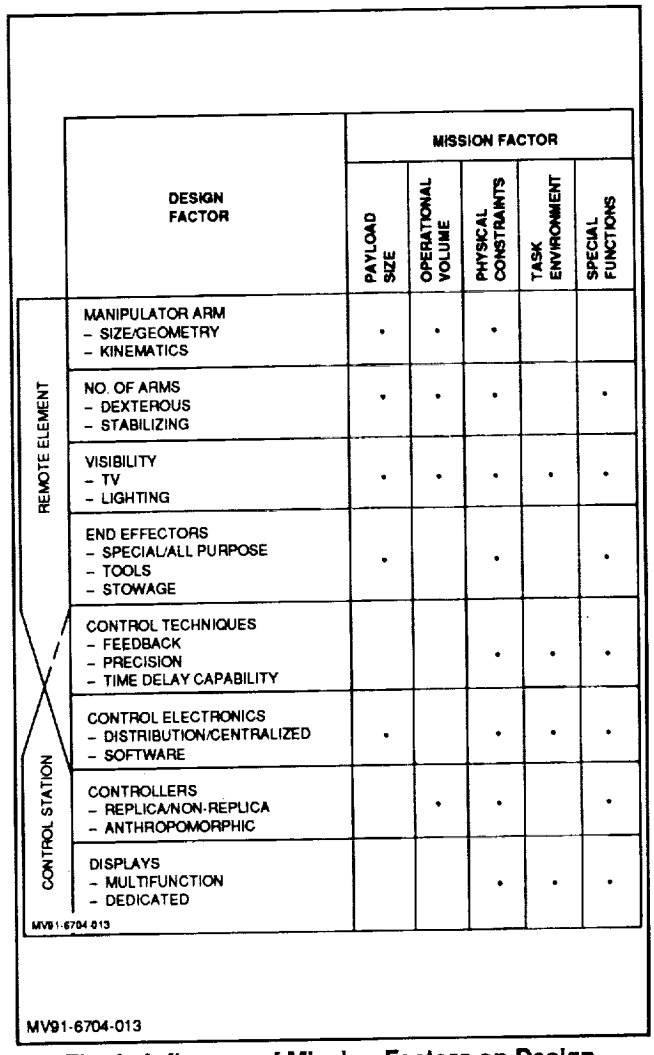


Fig. 9 Influence of Mission Factors on Design

ment effort to transition from skill-based operation to rule-based expert system operation to knowledge-based autonomous system operation (Fig. 12) to alleviate this bottleneck in the next decade.

Once the design requirements have been established and technology trades have been completed, the next step in the process is the resolution of identifed issues and the verification of the design concepts. The various methods to resolve issues are presented in Fig. 13 and the applicability of these methods to the issues discussed above is shown in Fig. 13A.

The numerical analysis is performed at the system specification level after functions have been assigned to hardware and software design elements. Analysis software such as Ascent Logic's Requirements Driven Design (RDD) -100 can be used to develop functional design requirements and contains function behavior models that can be linked into concept design segments for time-function execution analysis. This program is capable of developing performance time lines and computation and communications estimates for initial design verification and sizing exercises.

As the design definition matures, the graphical analysis and computation analysis tools are used to verify the per-

formance and constraints of the architectural elements. The graphical analysis tools such as IGRIP permit analysis of the geometry of the remote system in the work environment. It is an initial verification of geometric and kinematic performance of the mechanisms and task performance/time lines. It is especially useful in resolving remote system and worksite characteristics and compatibilities.

The computer simulation tools are used to develop the engineering performance of the system/subsystem elements. A typical set of tools includes analysis of weight, power, thermal, structural, and control systems. The simulation models are usually transfer function level models, with detailed design parameters, subjected to the stimulus of nominal/worst case environment parameters. The objective is to evaluate the element performances and verify the system budgets and performance allocations.

The hardware test/simulation phase is usually conducted at the full system level and utilizes engineering breadboard hardware/transform function models, prototype algorithms/software and environment mockups operated in real time. It is usually the first time human factors and hardware/software performance are evaluated in the operational environment as a total system. The decision to simulate at reduced scale or full scale depends on the fidelity of the interfaces

ISSUE AREA	ELEMENTS FOR RESOLUTION
MIX OF HUMANS & MACHINES	DECISION LEVELS HAZARDS EVALUATION LEVEL OF AUTOMONY/ PROCESSING LEVEL OF FEEDBACK/COMM
REMOTE SYSTEM CHARACTERISTICS	PHYSICAL SIZE & CAPACITY DEGREES OF FREEDOM END EFFECTORS & TOOLS VISION/LIGHTING
TASKSITE COMPATIBILITY	TERRAIN COOPERATIVE FEATURES WORK ARTICLES SAFETY
CONTROL STATIONS	CONTROLLER(S) DISPLAY(S) HUMAN FACTORS

MV91-6704-014

Fig. 10 Issue Areas

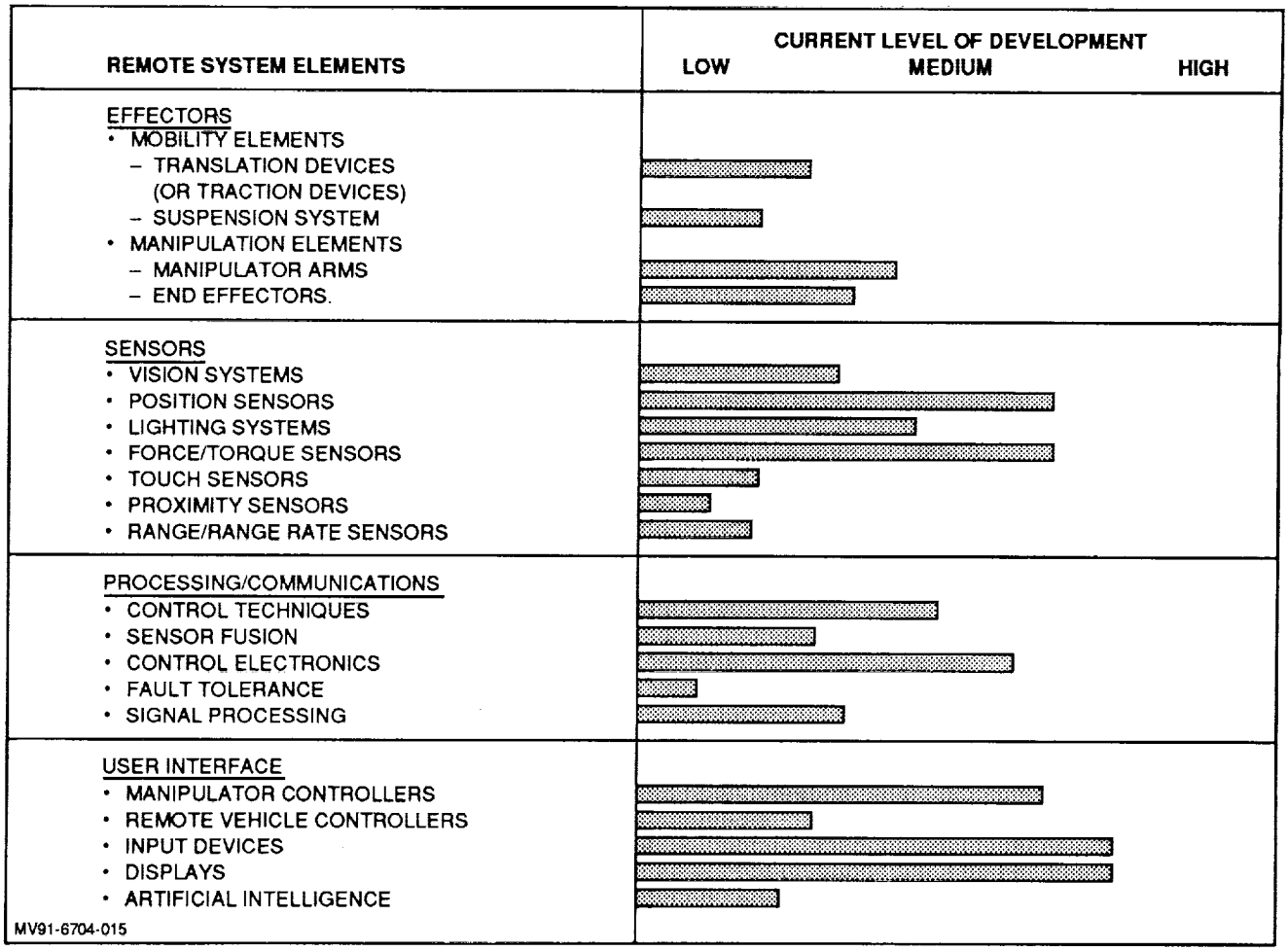


Fig. 11 Technology Requirements for Space Application

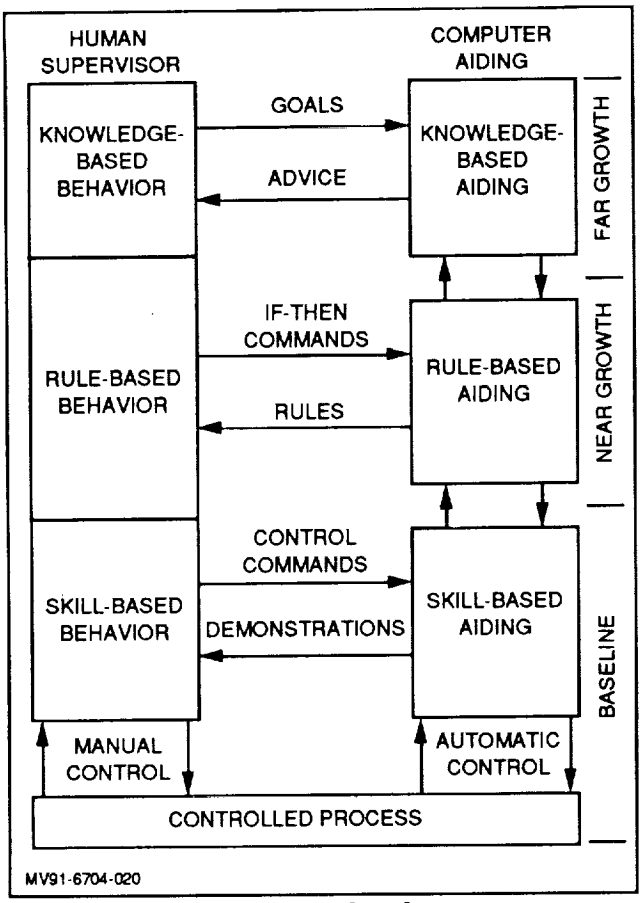


Fig. 12 Operator Interface

necessary to achieve system performance verification. Testing capabilities such as Grumman's Simulation Center are required; the Large Amplitude Space Simulator (LASS), the Advanced Space Workstation Lab and the Telerobotics Development Lab are needed for either full scale or scaled real time simulations for final remote system design concept verifications. The LASS contains a six-degree-of-motion device in a 50 ft x 50 ft x 20 ft high gaming area. The motion device can be programmed to emulate a space transport vehicle or a large scale robotic mechanism. The models that drive the motion device are real time dynamic kinematic models. The motion device is capable of supporting up to 1000 lbs (usually only the end effector of the vehicle or mechanism) at linear velocities of + 15 FPS and angular velocities of ±80 DPS. The gaming area can be provisioned with full scale mockups of the environment work articles. An additional 5 DOF motion can be added to test articles by mounting them on a Handling and Position Aid (HPA) capable of supporting 1000 lbs of payload. The HPA is a 5 DOF stiff, robust arm with morphology similar to the human arm. The applications of such approaches to resolve issues are discussed in the next section.

APPLICATION OF RESOLUTION APPROACHES

The issue resolution approaches discussed above are generally used to accelerate the development process while reducing the risk. Figure 14 shows how this resolution methodology can be used in an integrated fashion in the space hardware development process. In this section, we will show, by taking examples from various projects and the Grumman Telerobotics Development Laboratory, how these

METHOD	DEFINITION	ADVANTAGE	DISADVANTAGE
NUMERICAL ANALYSIS	COMPUTATION OF PERFORMANCE LEVELS FROM TASKS REQM'TS AND ENVIRONMENTAL CHARACTERISTICS	FAST, VERY ACCURATE, INEXPENSIVE	NOT ALWAYS FEASIBLE FOR COMPLEX ISSUES
GRAPHICAL ANALYSIS	GEOMETRIC STUDIES OF PHYSICAL ENVIRONMENT & CONCEPTS USING COMPUTER GRAPHICS	FAST, RELATIVE ACCURATE, MODERATELY EXPENSIVE	NOT ALWAYS FEASIBLE FOR INTRICATE SYSTEMS, SUBJECT TO SOME INTERPRETATION
COMPUTER SIMULATION	MATHEMATICAL MODELING OF PHYSICAL BEHAVIOR OF A SYSTEM	PRECISE FOR CERTAIN PROBLEMS, MODERATELY EXPENSIVE, EASY PARAMETER VARIATIONS	DEPENDENT ON FIDELITY OF MATH MODEL
HARDWARE TEST/SIMULATION-REDUCED SCALE	USE OF REPRESENTATIVE REDUCED SCALE PHYSICAL MODELS IN CONJUNCTION WITH COMPUTER MODELS	REALISTIC OPERATIONS, SOME HARDWARE EFFECTS, HUMAN FACTORS INCLUDED	MODERATELY COSTLY, POTENTIAL SCALING ERRORS
HARDWARE TEST/SIMULATION-FULL SCALE	USE OF REPRESENTATIVE FULL SCALE MODELS OF SYSTEM ELEMENTS IN CONJUNCTION WITH COMPUTER MODELS	CLOSEST TO REAL OPERATIONS, INCLUDES HARDWARE NONLINEARITIES, HUMAN FACTORS INCLUDED	COSTLY, LONGER SCHEDULE TIME, LIMITED PARAMETER VARIATIONS

MV91-6704-016

Fig. 13 Issues Resolution Methods

ISSUE AREA	ELEMENTS FOR RESOLUTION	RESOLUTION APPROACHES				
		NUMERICAL ANALYSIS	GRAPHICAL ANALYSIS	COMPUTER SIMULATION	HARDWARE TEST/SIMULATION -REDUCED SCALE	HARDWARE TEST/SIMULATION -FULL SCALE
MIX OF HUMANS & MACHINES	DECISION LEVELS	•	•	•	•	•
	HAZARDS EVALUATION		•	•	•	•
	LEVEL OF AUTONOMY/ PROCESSING	•		•		•
	LEVEL OF FEEDBACK/COMM	•		•		•
REMOTE SYSTEM CHARACTERISTICS	PHYSICAL SIZE & PERFORMANCE	•	•	•	•	•
	DEGREES OF FREEDOM		•		•	•
	END EFFECTORS & TOOLS		•	•	•	•
	VISION/LIGHTING		•		•	•
TASKSITE COMPATIBILITY	TERRAIN		•		•	•
	COOPERATIVE FEATURES	•	•	•		•
	WORK ARTICLES					•
	SAFETY		•	•	•	•
CONTROL STATIONS	CONTROLLER(S)		•	•	•	•
	DISPLAY(S)		•		•	•
MV91-6704-017	HUMAN FACTORS		•		•	•

Fig. 13A Resolution Approaches

approaches have been used in the remote system development process.

Graphical and Computer Analysis

Math models of candidate remote systems elements, such as a manipulator arm, can be developed using data from automated drafting programs. Using a Silicon Graphics high-resolution, three-dimensional workstation and a Deneb Robotics software package IGRIP, a solid model with moving joints can be developed, as shown in Fig. 15 for a 3-arm capture mechanism for a tumbling satellite retrieval system, to determine task feasibility (e.g. collision potential) and design viability (e.g. reach requirements). It provides real-time, kinematic simulation of sufficient fidelity to support task scenario development, operator interface and procedures training, system kinematics display, camera views, and realistic operations. Candidate control algorithms from control systems analysis programs such as PROTOBLOCK can be input into the IGRIP simulation for evaluation.

Another example of graphical modeling of a concept is presented in Fig. 16, which shows a robot performing a servicing operation on a space platform. Such modeling

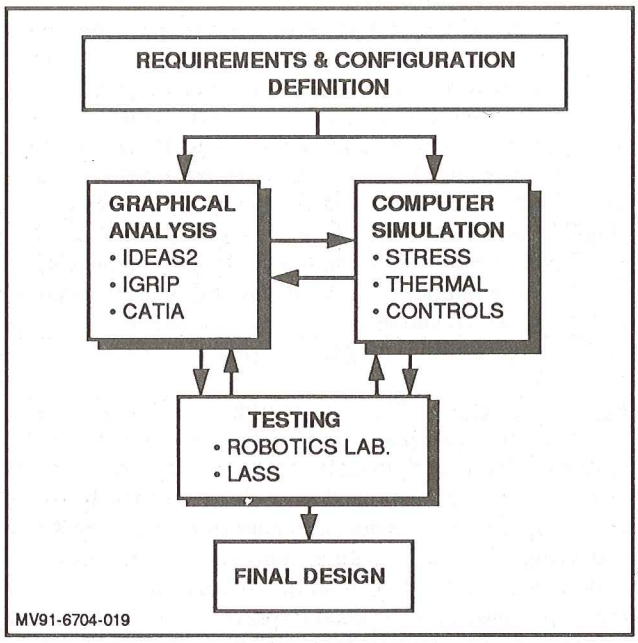


Fig. 14 Integration of Issue Resolution Methodologies into a Program

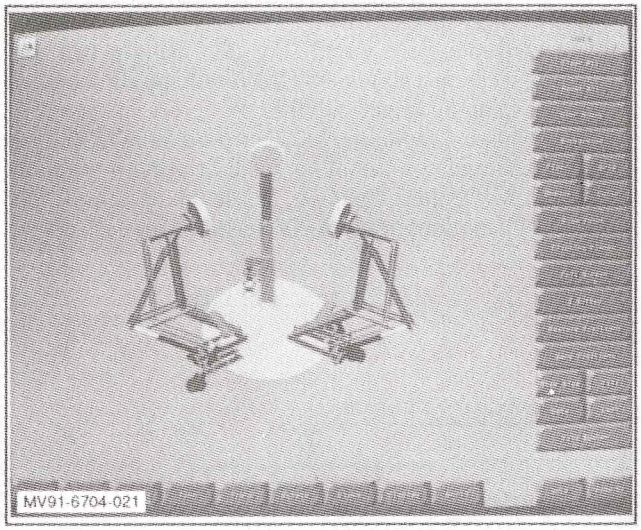


Fig. 15 3-Arm Capture Kit IGRIP Solid Model

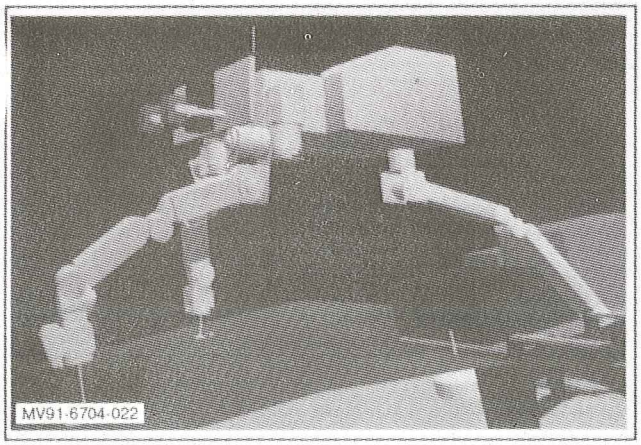


Fig. 16 Typical IGRIP Robot Scenario

analysis, when performed prior to initial hardware fabrication, enables lessons learned to be applied during the design phase, before commitments to hardware are made. The available computerized, geometrical data base can also then be transformed into finite element models for dynamic and thermal analyses.

Testing

Testing of remote systems, in full and partial scale, can be described using four general categories: large robots, small robots, remote vehicles, and EVA astronaut/robot operations as further described below. All tests involve interfaces with and direction from a control station. In general, the testing results in measures of system performance such as task times, identification of design improvements, and suggested changes to the task environment.

Large Robots - Large robots, i.e. manipulator arms significantly longer than 10 ft with associated sensors, in conjunction with a control station, can be tested using computer simulation approaches and actual test articles. The shuttle's Remote Manipulator System (RMS) has been tested using a simulation approach in Grumman's Large Amplitude Simulator (LASS) as shown in Fig. 17 for placement of a plasma diagnostics package on a spacelab pallet. A dynamics and kinematic model of the RMS has been installed in the LASS. This enables the RMS end effector motion to be emulated by the LASS platform. A functional block diagram of the facility as it is utilized for simulation of the RMS is illustrated in Fig. 18. The RMS software and dynamics are contained in the hybrid computer. It receives its input commands from the rotational and translational hand controllers located in the shuttle aft flight deck control station mockup in the laboratory (Fig. 19).

In another simulation (Fig. 20) a space erectable radiator element was inserted into a receptacle by the RMS. In this test, the operator was able to to successfully insert a simulated 50 ft radiator test article in an evaporator slot with a vertical clearance of 0.25 inch and horizontal clearance of ± 1 inch. The test runs were made with a tactile sensor which enabled the operator to detect a touch before the force built up to 5 lbf. An augmented grapple target and an enlarged slot opening (±1 in both directions) enabled 40 test runs to be completed with only 6 recorded touches.

Testing of a large robot using a test article is exemplified in Fig. 21 by the handling and positioning aid (HPA) which was a candidate manipulator for moving and holding large, heavy payloads within the servicing envelope of the shuttle payload bay. In this case, the HPA with a simulated snare end effector and TV camera is shown aligning to a grapple fitting with control from a remote console.

Small Robots - Small robots, i.e. one or more arms generally less than 10 ft long with associated control stations, can be tested using computer graphics and test article. Fig. 22 illustrates the Deneb IGRIP graphics, discussed above, used to simulate a two-armed robot which is being controlled by two 6-DOF hand controllers.

Early testing using a pair of master/slave manipulators is shown in Fig. 23 with the operator performing a simulated satellite servicing task. Simulations of small robots carried by a large robot have been performed using the LASS platform which now creates a positioning system simulation with its 6 DOF capability. The RCS module changeout task, as performed telerobotically on the LASS, is shown in Fig. 24. The subtasks included attaching the protective covers over the nozzles, attaching a tether, untightening the fasteners, extracting the module, and installing the replacement module. The object being manipulated must also be offloaded to work properly within the design capacity of the robots; this also provides the capability for "zero-g" simulation.

A more advanced two-arm robot (Fig. 25), each arm having 7-DOF, which also has an additional 3-DOF "torso" for a total of 17 DOF, is another test article which can be controlled by a pair of mini-master controllers as shown in Fig. 26. These controllers can be used to evaluate limited operating volume applications, e.g. within a one ft cube (Fig. 26) and to evaluate performance with candidate displays (Fig. 27). This robot has been used to investigate tasks requiring a capability to reach around obstacles such as truss structure (Fig. 28) and then use a camera installed close to the end effector to perform inspections of a truss joint as shown in Fig. 29. Another example of the use of such a test article is shown in Fig. 30 which shows an off-loaded hydrazine fuel coupling being installed using the two arms.

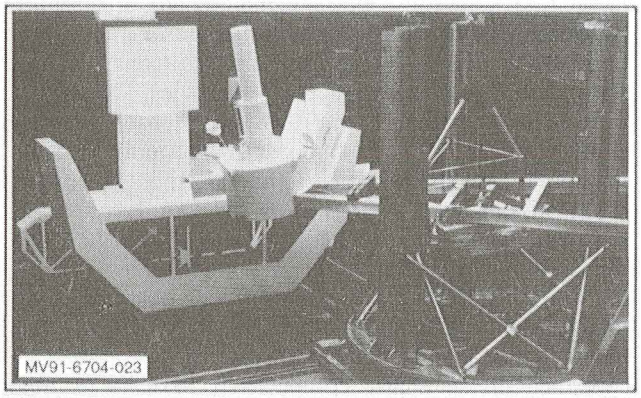


Fig. 17 Experiment Simulation on LASS

Remote Vehicles - We view remote vehicles as vehicles which translate on a planetary surface or in space, controlled remotely. The LASS, in conjunction with appropriate control consoles, has been used to simulate free flying spacecraft. Fig. 31 shows a special inspection vehicle, the Maneuvering TV (MTV) with a snare end effector, as a half scale model, on the LASS simulator aligning to capture a mockup of the Long Duration Exposure Facility spacecraft. The motion of the Orbital Maneuvering Vehicle (OMV) has also been simulated, including the capture of a satellite with a three-point docking ring shown separately in Fig. 32 and on a full scale OMV mockup in Fig. 33. Early tumbling satellite capture concepts were tested (Fig. 34) using a two arm capture concept on a simulated OMV with the HPA used to position a spinning target satellite.

This combined capability has been used more recently to simulate capture of a tumbling satellite using a specially designed OMV front end kit (Fig. 35). In this case, the LASS simulated all motions of the OMV, based on inputs from the

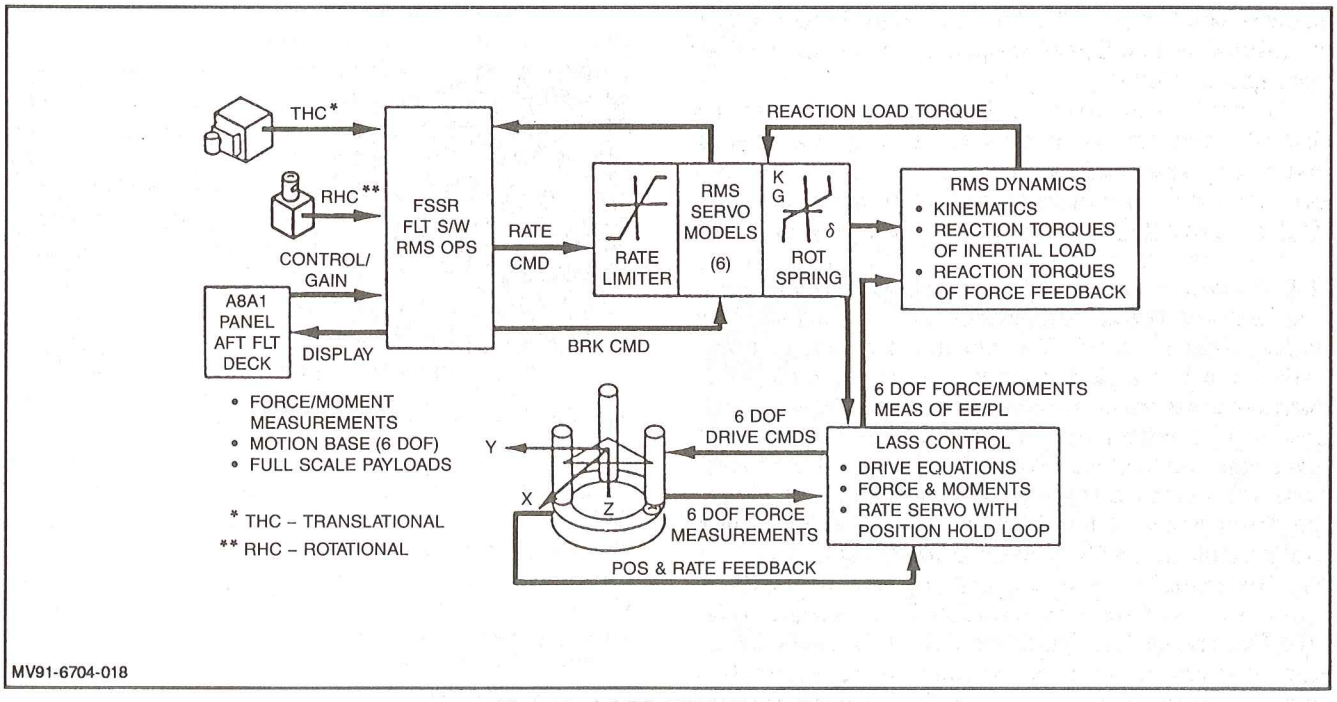


Fig. 18 LASS RMS Block Diagram

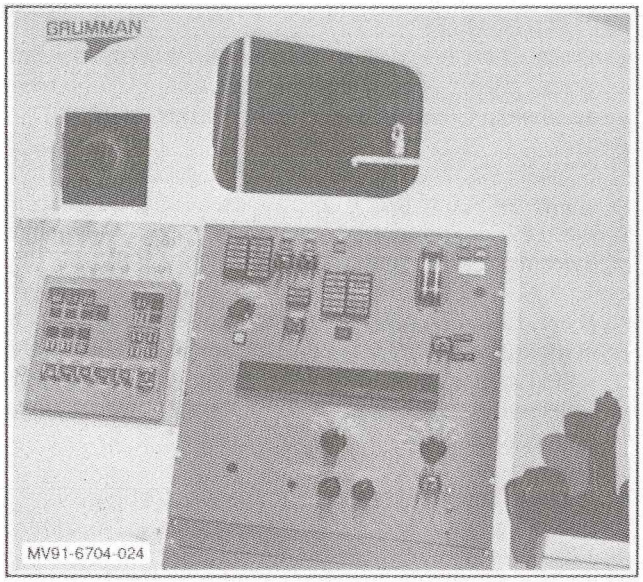


Fig. 19 Shuttle RMS Control Station Mockup

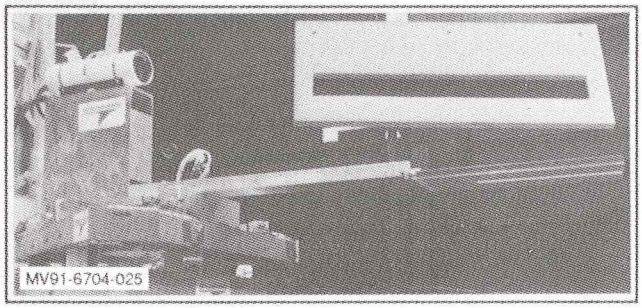


Fig. 20 Space Radiator Insertion Test on LASS

remote control console. The kit, a three dual-link arm device installed on the LASS platform, performed the capture of the satellite being held by the HPA.

A control console from which such teleoperation mission feasibility tests are performed is shown in Fig. 36. Controls and display graphical aids added to the console enhance operator human factor performance of the mission.

EVA Astronaut/Robot Operations - EVA astronaut/robot operations refer to the special case of extra vehicular activity (EVA) where astronauts operate in close proximity to robots, and is broken out as a separate category because of the safety issues involved. The Manipulator Foot Restraint (MFR) which is a platform at the end of the RMS which supports an astronaut is one example. The MFR was developed by Grumman using the LASS as a development tool and a scale model (Fig. 37) based on functional and operational requirements. This model and task simulations led to the development of test hardware (Fig. 38), which was evaluated on the LASS by suited astronauts (Fig. 39). With the MFR on the LASS simulating the dynamics of the RMS, a wide variety of servicing scenarios were simulated (Fig. 40). This testing has significantly aided the resolution of safety concerns which otherwise would not have been identified early in the design process. In addition to evaluation of astronaut tools, equipment and tasks, the RMS control

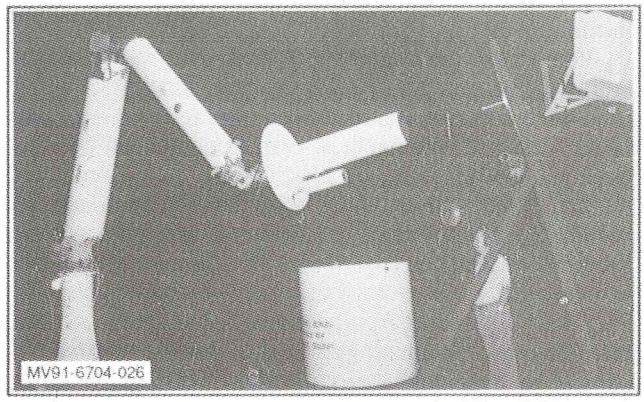


Fig. 21 Moving Heavy Payloads with HPA

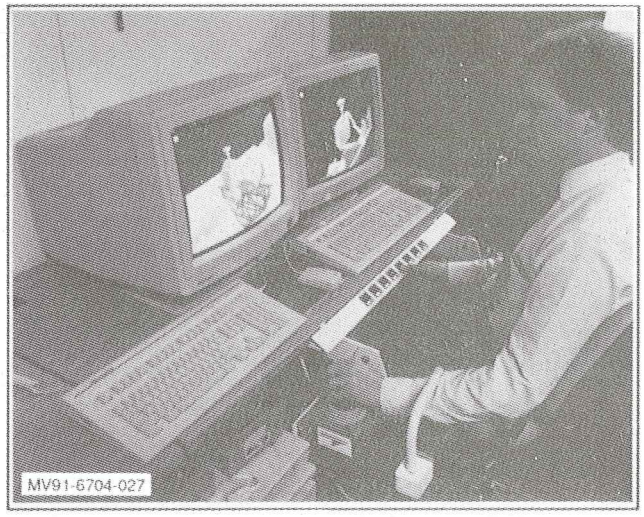


Fig. 22 Simulations Using a Workstation

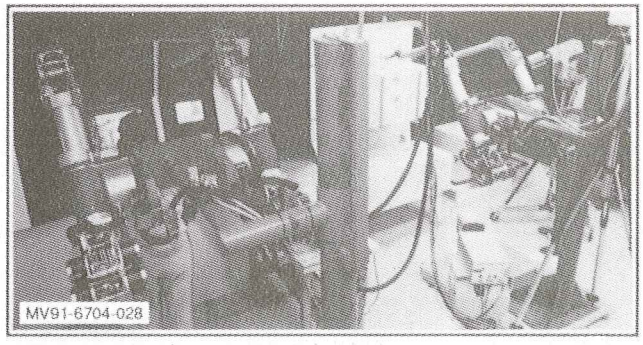


Fig. 23 Simulations Using Master/Slave Manipulators

system interaction with a suited astronaut was evaluated. Soft and breakaway mock-ups were used to prevent damage to equipment or injury to test subjects in case of an unchecked runaway. The further use of scale models is shown in Fig 41 which shows a small robot operating in conjunction with an EVA astronaut on a MFR.

The use of extensive testing of the types discussed above are essential in the development of satisfactory, low-risk remote systems.

Summary & Conclusions

Future space activities are expected to rely more and more on remote systems. These systems and the technology used have counterparts in many future terrestrial applications. Their development is best accomplished by using graphical analysis and test articles early in the process to resolve design issues before commitments are made to a design. Such issues have been discussed and the methods of resolution have been described with examples given based on many years of remote systems development experience. The "lessons learned" from such techniques contribute to the identification of technology advances, accelerate the development process, and lower the risks associated with the eventual final design.

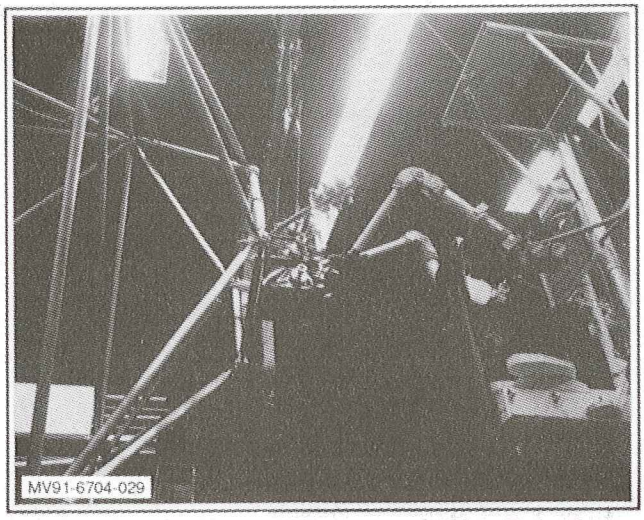


Fig. 24 RCS Module Changeout Using Robotics

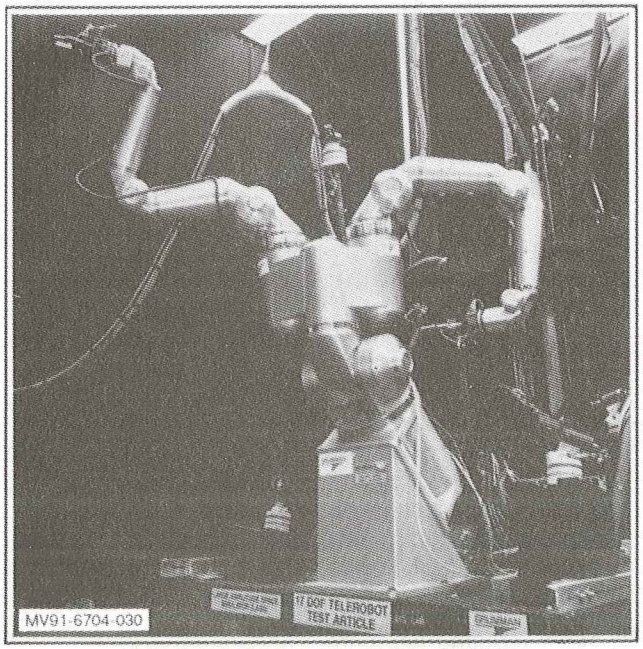


Fig. 25 17-DOF Laboratory Robot

BIBLIOGRAPHY

- Olsen, R.E., "Dexterous Manipulator Laboratory Program." Paper presented at the 18th Annual Conference on Manual Control, Dayton, OH, 10 June 1982.
- Schaefer, R.H. et al, "The Roles of Astronauts & Machines for Future Space Operations." Paper presented at the 15th Intersociety Conference on Environmental Systems, San Francisco, CA, 15-17 July 1985.
- Byler, E. & Olsen, R.E., "Automation & Robotic Technologies for Commercial Space Factories." Paper presented at the SPACE-88 Conference on Engineering, Construction and Operations in Space; Albuquerque, NM, August 1988.
- "Analysis of Remote Operating Systems for Space-Based Servicing Operations." Grumman Contract No. NAS9-17066.
- Schaefer, O., "Simulation Results of a Remotely Operated Tumbling Satellite Retrieval (TSR) Kit." Paper presented at the AIAA Space Programs & Technologies Conference, Huntsville, AL, 25-28 September 1990.
- Olsen, R.E. et al, "Grumman Capabilities: Large Amplitude Space Simulator (LASS)." Paper presented at the SAE '87 Conference, 6-7 October 1987.
- "Flight Telerobotic Servicer, Phase B." Grumman Contract No. NAS 5-30248.
- Albus, J., H. McCain & R. Lumia "NASA/NBS Standard Reference Model for Telerobot Control Systems Architecture, NBS Tech. Note 1235." June 1987.

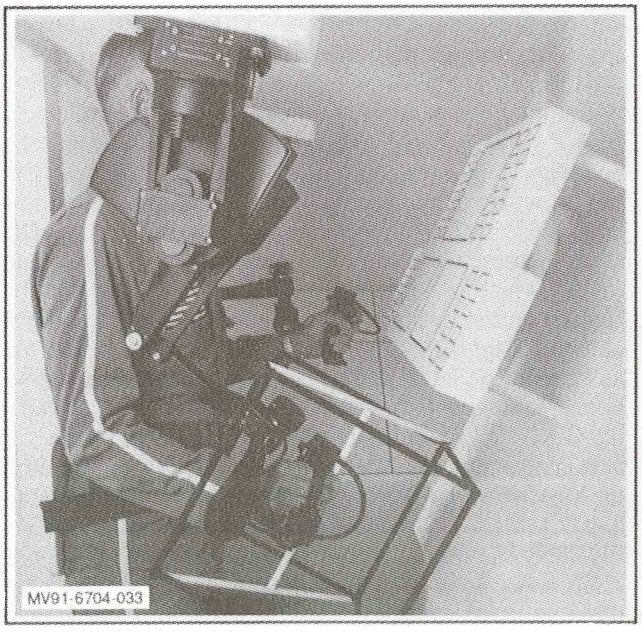


Fig. 26 Simulations with Mini-Masters

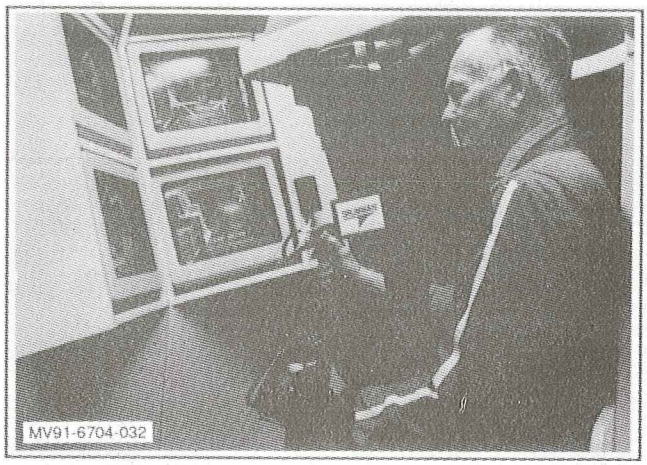


Fig. 27 Display/Task Performance Evaluation

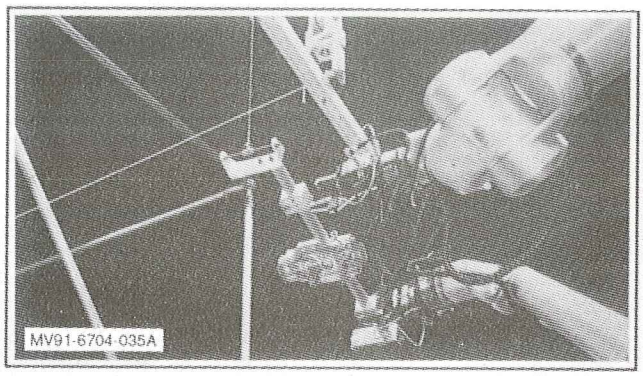


Fig. 30 Robotic Fuel Coupling Installation

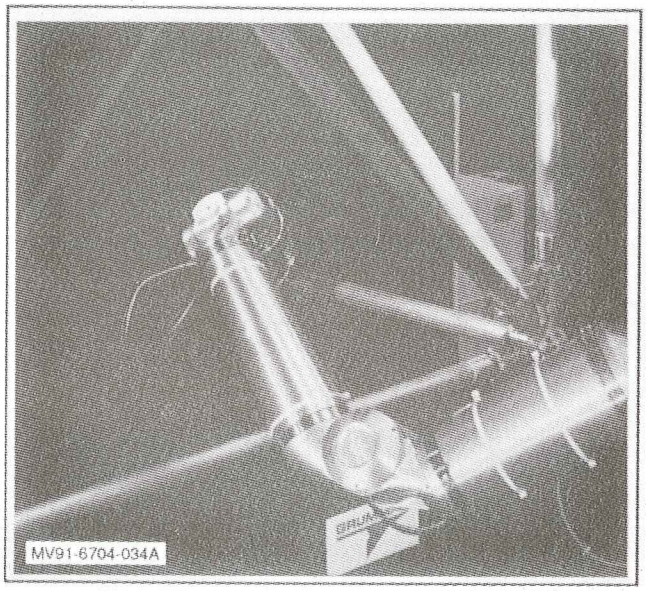


Fig. 28 Joint Inspection Around Obstacles

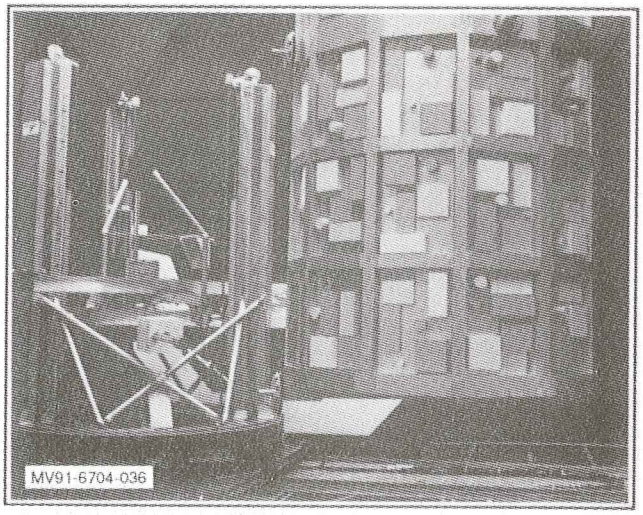


Fig. 31 LDEF Inspection Simulation with MTV

Fig. 29 Inspection – Wrist Camera View

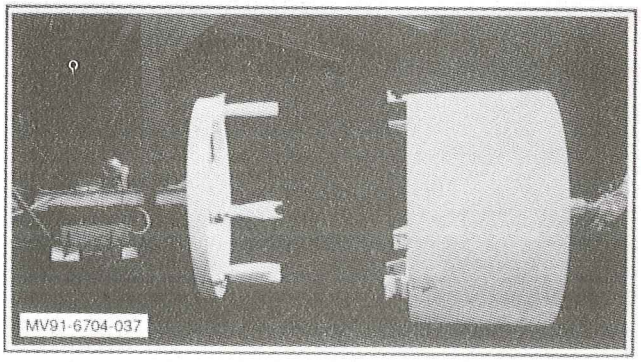


Fig. 32 3-Pt. Docking Mechanism Simulation

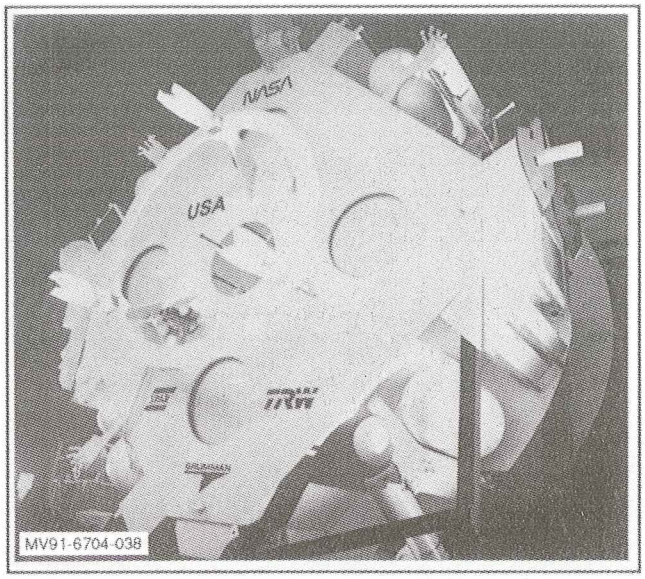


Fig. 33 Full Scale Mockup of OMV/3-Pt. Mechanism

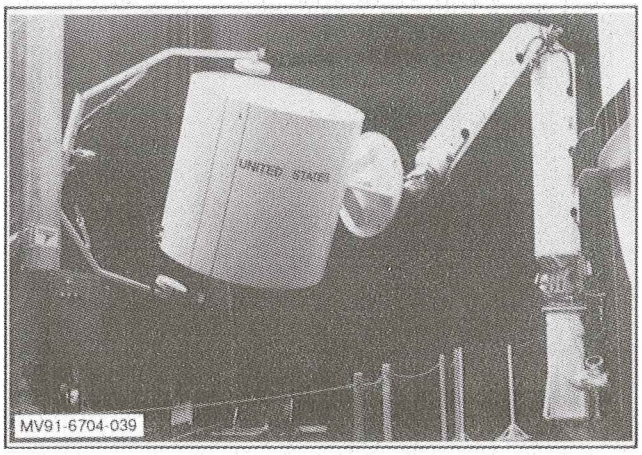


Fig. 34 Early TSR Capture Simulation on LASS

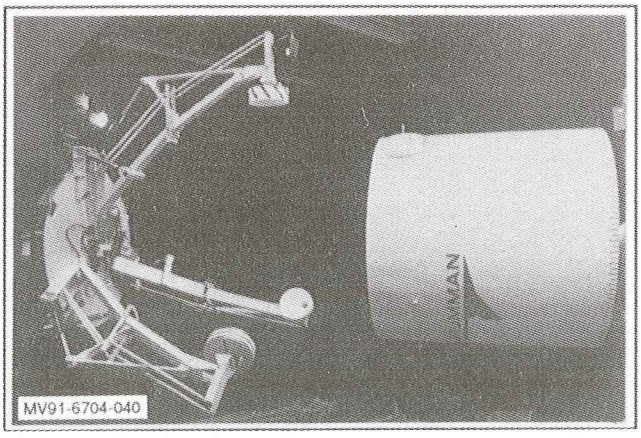


Fig. 35 3-Arm Kit Capture Simulation on LASS

Fig. 36 OMV/Front End Kit Control Console

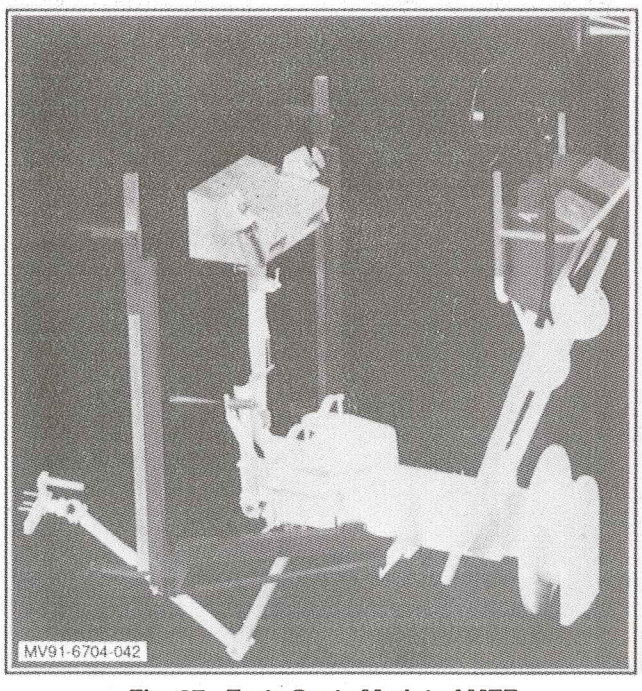


Fig. 37 Early Scale Model of MFR

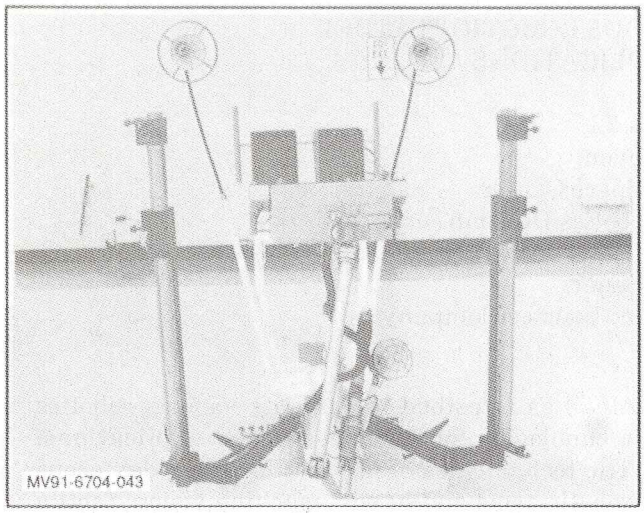


Fig. 38 MFR Development Test Article

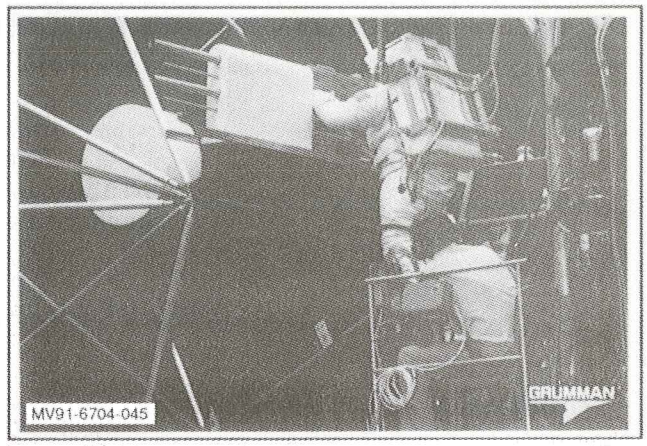


Fig. 40 Thermal Bus Assembly Simulation

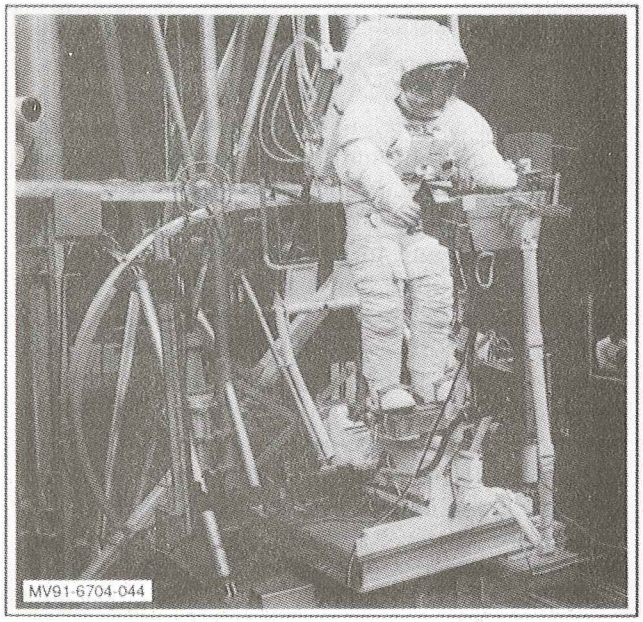


Fig. 39 Astronaut MFR Simulations on LASS

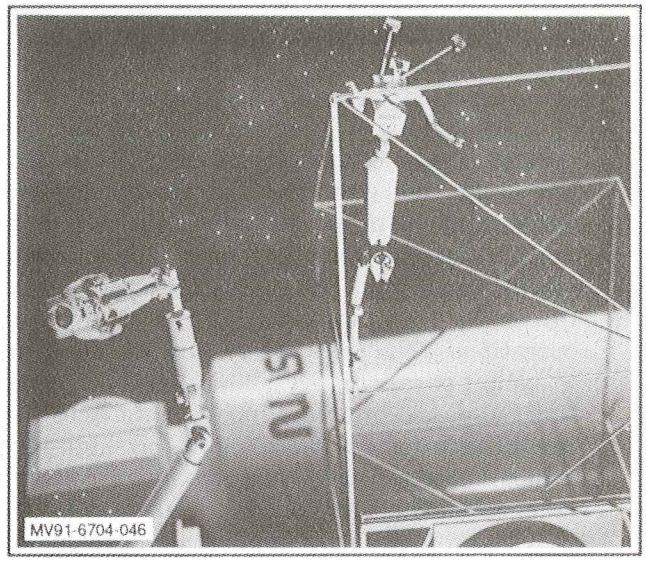


Fig. 41 Combined Astronaut/Robot Operations

AN INTEGRATED DEXTEROUS ROBOTIC TESTBED FOR SPACE APPLICATIONS

Larry C. Li
Hai Nguyen
NASA/Johnson Space Center
Automation and Robotics Division

Edward Sauer
Lockheed Engineering and Science Company

ABSTRACT

An integrated dexterous robotic system was developed as a testbed to evaluate various robotics technologies for advanced space applications. The system configuration consisted of a Utah/MIT Dexterous Hand, a PUMA 562 arm, a stereo vision system, and a multiprocessing computer control system. In addition to these major subsystems, a proximity sensing system was integrated with the Utah/MIT Hand to provide capability for non-contact sensing of a nearby object. A high-speed fiber-optic link was used to transmit digitized proximity sensor signals back to the multiprocessing control system. The hardware system was designed to satisfy the requirements for both teleoperated and autonomous operations. The software system was designed to exploit parallel processing capability, pursue functional modularity, incorporate artificial intelligence for robot control, allow high-level symbolic robot commands, maximize reusable code, minimize compilation requirements, and provide an interactive application development and debugging environment for the end users. This paper presents an overview of the system hardware and software configurations, discusses implementation of subsystem functions, and recaps lessons learned from our work. Current work and future evolution of the system are also discussed.

INTRODUCTION

An integrated dexterous robotic system was developed by the NASA Johnson Space Center (JSC) as a testbed to evaluate various robotics technologies for advanced space applications. The technologies of interest include: dexterous robotic arms and hands, machine vision, tactile and proximity sensing, grasping and manipulation algorithms, parallel computational architecture, and artificial intelligence. The configuration of the testbed system consisted of a 16 degrees-of-freedom (DOF) Utah/MIT Dexterous Hand, a 6 DOF PUMA 562 arm, a stereo vision system, and a VMEbus-based multiprocessing control system. In addition to the position and force sensors already present in the Utah/MIT Hand, an 8-element proximity sensor system was developed and integrated with the hand to provide near-range non-contact sensing capability. A high-speed fiber-optic link, also developed at JSC, was used to transmit digitized proximity sensor signals back to the multiprocessing control system. A hierarchical functional architecture was implemented to provide serial-parallel execution of limb motions. A JSC-developed expert system tool called the C Language Integrated Production System (CLIPS) was used as a rule-based robot programming environment. The system configuration allows autonomous operation and teleoperation. The purpose of this paper is to present an overview of our implementation. First, background and other work in similar areas are reviewed. Then the objectives for developing this testbed system are stated. The system overview covers subsystem implementations. Examples of robot programming are given in the section, Robot Programming Using CLIPS. Finally, the last section summarizes lessons learned from our work.

BACKGROUND

NASA has been active in robotics from the early days of the space program. The Viking mission to Mars is one shining example. Recent progress in robotics technology has allowed NASA to design robots that will help to increase productivity in space. These space robots may be used to perform dangerous or laborious tasks which otherwise would have to be performed by the astronauts. The Shuttle Remote Manipulator System (SRMS), for example, has demonstrated its effectiveness in on-orbit satellite retrieval and repair. The Flight Telerobotic Servicer (FTS), targeted for the Space Station, will be the first advanced multi-function robot in space. The Extravehicular Activity (EVA) Retriever, currently under development at JSC, is an advanced space robot designed for short-range, contingency retrieval and rescue missions. Other advanced robotic systems under development by NASA, such as the Lunar/Mars Rover and the Satellite Servicing System, are other examples of NASA's commitment to further enhance and apply the robotics technology.

The Automation and Robotics Division at JSC has developed a dexterous robotic testbed to develop and evaluate various enabling robotics technologies for space applications. Different from other NASA-developed robotic systems, this testbed emphasizes the development, integration, and application of dexterous robotic hands and arms. Outside NASA, there are many other research efforts with similar emphasis and interests. Clark and Demmel, et. al,[3] has developed a dexterous robotic system consisting of a Utah/MIT Hand, a PUMA 560 arm, a Polhemus 3D Tracker, and a VPL Data Glove. Their implementation was optimized for teleoperation with the VPL Data Glove and the Polhemus 3D Tracker providing position control of hand and arm, respectively. Allen, Michelman, and Roberts[1] described an integrated system for dexterous manipulation. Their implementation also included a Utah/MIT Hand and a PUMA 562 arm. Although similar to the implementation described in reference 3, Allen, Michelman, and Roberts developed their system for autonomous operations. Allen and Roberts[2] have successfully demonstrated haptic object recognition using this system. They have integrated tactile sensors with the Utah/MIT Hand, and used a descriptive language called DIAL for high-level control. Narasimhan, in his master's thesis[9], described a VME multi-processing control system for the Utah/MIT Hand. His system contained several 68020 computer processing units (CPUs) for servo-level and task-level controls. A real-time operating system called Condor was integrated with the system to provide timing, task scheduling, and other process control functions. Salisbury, Brock, and O'Donnell[11] built their dexterous hand system around the Stanford/JPL Hand. They introduced the usage of a LISP machine as the high-level controller providing rule-based programming capability. Stansfield[12,13] developed a dexterous arm/hand system with knowledge-based visually-guided grasping. Our implementation incorporated and enhanced some of the features and concepts found in these research efforts.

OBJECTIVES

Our overall objective was to develop, evaluate, demonstrate, and enhance dexterous robotics technologies for space applications. Although our overall objective was somewhat general, it can be broken down into three specific goals: (a) develop and demonstrate capabilities of dexterous robotic systems for space applications; (b) investigate, implement, and evaluate latest advanced robotics technologies; and (c) develop an integrated testbed to support and demonstrate autonomous operation and teleoperation of dexterous robotic systems. Rationales behind these three objectives are explained next.

a. Develop and demonstrate capabilities of dexterous robotic systems for space applications

Future space robots are required to be highly versatile and productive. In order to achieve the required versatility and productivity, these robots should be equipped with intelligent dexterous arms and hands to handle a

multitude of tasks. Because most dexterous robotic hands are modeled after the human hand, their anthropomorphic designs allow the robots to share a common set of tools and handholds with the astronauts, thus minimizing any redesign of existing flight hardware. Other dexterous robotic components such as robotic arms with redundant DOF are also important in providing robots with multiple trajectory options for collision avoidance and path planning.

b. Investigate, implement, and evaluate latest advanced robotics technologies

Besides dexterous robotic hands and arms, other advanced robotics technologies such as parallel computers, machine vision, tactile and proximity sensing, expert systems, and neural networks are also important in the development of an intelligent space robot. Since there is usually an appreciable time gap between emergence of a new technology and the actual implementation of this technology on a flight system, it is important for NASA to keep up with the latest advanced robotics technologies so that the space robots will not be technologically outdated.

c. Develop an integrated testbed to support both autonomous operations and teleoperations

In order to evaluate and demonstrate technologies mentioned above, a testbed system must be developed. It is important for this testbed to be fully integrated so that different technologies can work together to achieve the high-level functions required in an intelligent space robot. An integrated testbed system will also allow us to verify planned operations and identify any unanticipated problems. The testbed system should support both autonomous operations and teleoperations, since both modes of operation are likely to find applications in future space activities.

SYSTEM OVERVIEW

Based on the objectives discussed in the previous section, a testbed system was established. Figures 1a and 1b show the current Smart Hand testbed system configuration. This section provides an overview of the system, which includes descriptions of hardware system architecture, software system architecture,

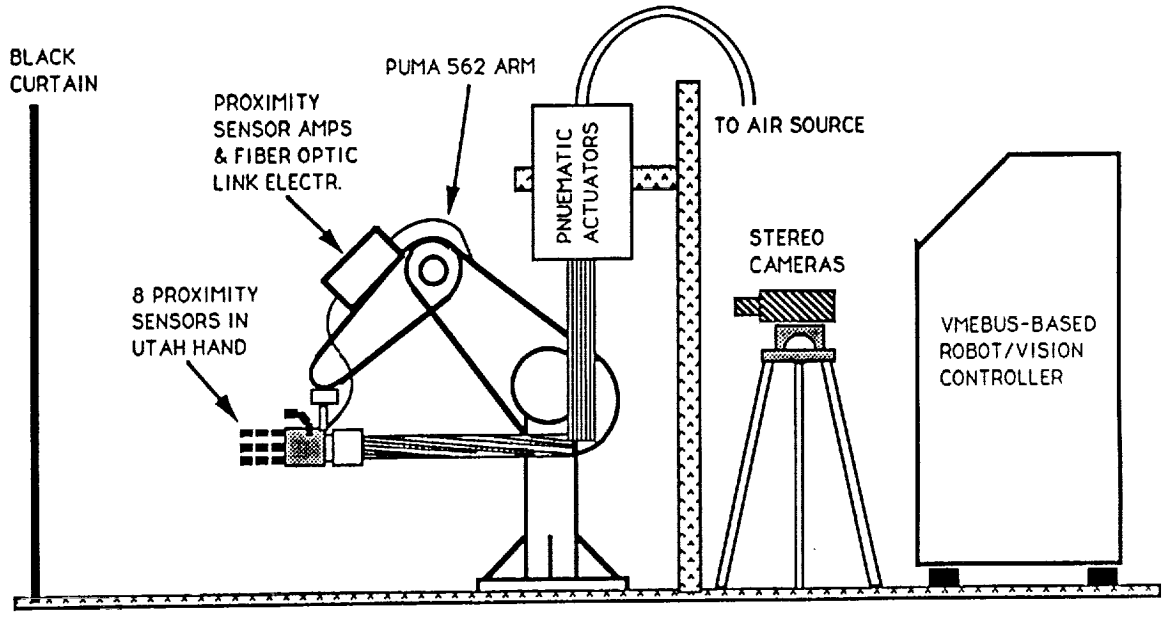


Figure 1a. Current Smart Hand testbed system configuration.

dexterous arm and hand subsystem, vision subsystem, teleoperator interface, and the proximity sensor subsystem.

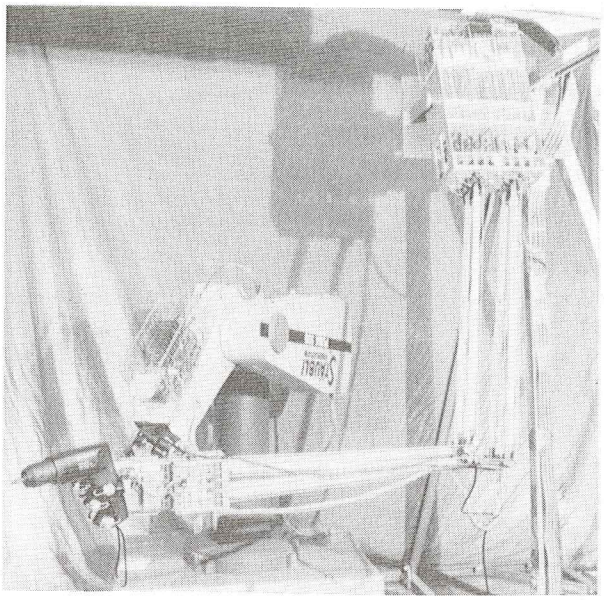


Figure 1b. Arm and Hand of the testbed system.

Hardware System Architecture

Major hardware components in the system include a Utah/MIT Hand, a PUMA 562 Arm, a stereo vision system, an EXOS Dexterous Hand Master, infrared proximity sensors, and a computer control system. The computer control system is a multiprocessing system with three 68020 single board computers (SBC) mounted inside a 20-slot VMEbus chassis. Each SBC is outfitted with pSOS™ - a real-time multi-tasking operating system kernel, and pRISM™ - a multiprocessing operating system, both of which are products of Software Components Group. Figure 2 shows the functional block diagram of the current system configuration. Each 68020 SBC is responsible for a specific task such as arm control, hand control, and vision control. The software running on these SBCs are discussed in greater detail in the Dexterous Arm and Hand Subsystems and Teleoperator Interface sections. A multi-function VMEbus system controller board arbitrates bus access among the SBCs. The serial port on-board the system controller provides the communication channel to the Unival PUMA arm controller.

Two Data Translation video frame grabbers are used to capture images from the two cameras. The digitized images are processed by the vision controller to produce a 3D vector pointing at the target. This vector is sent to the robot arm and hand controllers for reaching and grasping. The analog-to-digital (A/D) and digital-to-analog (D/A) converters are used to interface with both the Utah/MIT analog controller and the EXOS Dexterous Hand Master. A parallel digital input/output (I/O) board connected with a high-speed fiber-optic link gathers data from the proximity sensor subsystem. Fiber-optic transmission is used to avoid electromagnetic interference (EMI) generated by the large motors located inside the PUMA arm, and to reduce the number of wires bundled at the arm joints. A VMEbus-based 386SX personal computer (PC) is embedded inside the chassis for two purposes: (1) to provide a software development environment for the system programmers, and (2) to host an intelligent rule-based system for developing applications at the symbolic level. These two functions correspond to two phases of operation: development phase and operational phase.

During the development phase, the PC operates under MS-DOS. A 68020 C cross-compiler is used to generate executable codes from user-written source codes. The executable codes are then loaded, via the VMEbus backplane, into the SBC's dual-ported memories. During this time, the SBCs are in idle, waiting for signals to begin execution. During the operational phase, all subsystem software is loaded and started. The 386SX PC begins executing an expert system shell, the CLIPS. The CLIPS communicates with the subsystem SBCs through the *system executive*. All applications are developed under this shell, using CLIPS and user-defined syntax.

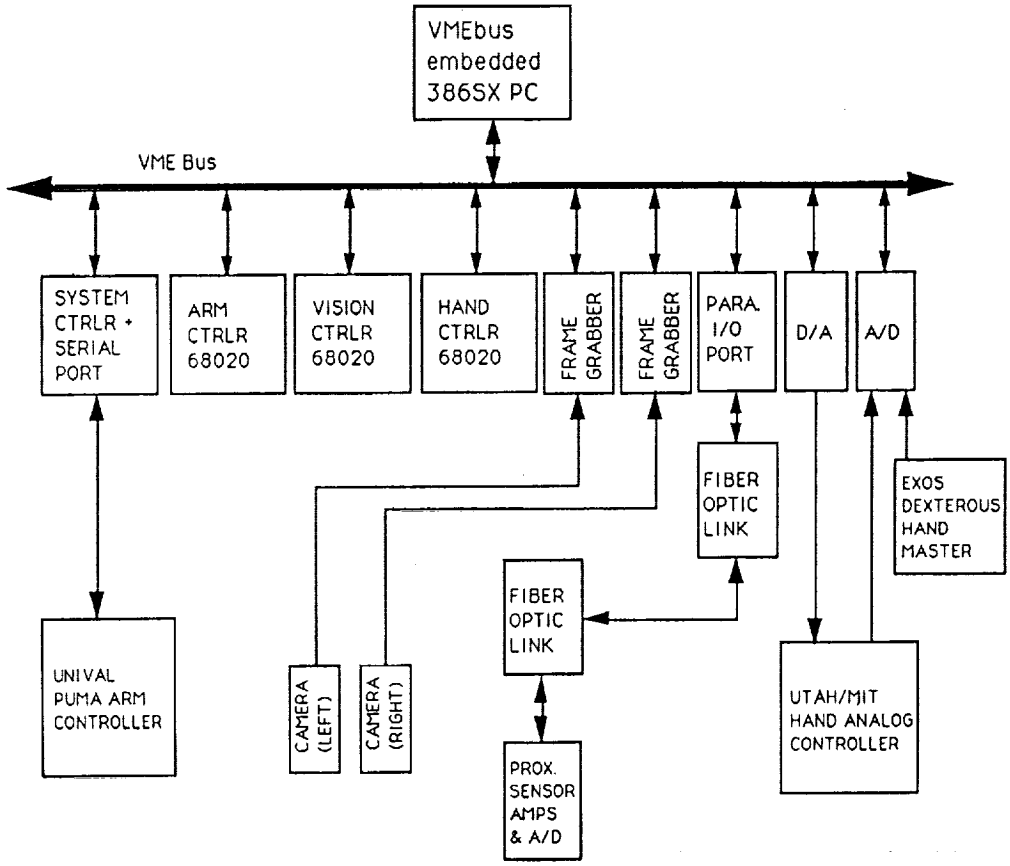


Figure 2. Functional block diagram of the current system configuration.

Software System Architecture

The software system architecture is designed based on the following objectives: (1) exploit parallel processing capability, (2) pursue functional modularity, (3) incorporate artificial intelligence for robot control, (4) develop high-level, symbolic robot commands, (5) maximize reusable code, minimize compilation requirement, and finally, (6) provide an interactive application development and debugging interface. Figure 3 illustrates the software system architecture. Software for the top three layers are running on the 386SX PC, while software for each subsystem in the subsystem layer are running on separate SBCs.

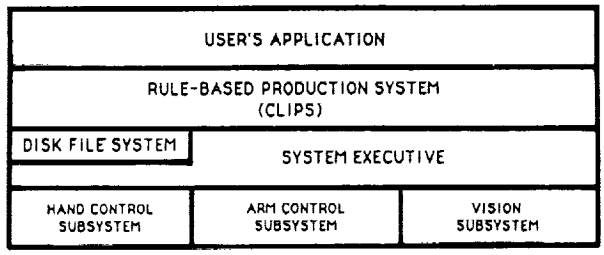


Figure 3. Software system architecture.

Robot arm and hand trajectories are *serial* executions of *parallel* motions. For example, the action of grasping for a ball consists of the following steps:

a. Locate the ball
b. Reach for the ball while opening hand
c. Close hand around the ball
d. Retract arm and hand

Steps *a*, *b*, *c*, and *d* must be executed *serially*, otherwise the task will not be accomplished properly. However, within each serial step, parallel motion takes place. Take Step (*b*) for example, the hand opens while the arm reaches for the ball. During opening of the hand, each finger joint should move concurrently; otherwise the motion would be awkward and time consuming. Understanding this principle, the software architecture was designed to realize *serial-parallel* motions. The overall architecture contains four layers: *user application*, *rule-based production system*, *system executive*, and *subsystem layer*. There are three functional subsystems executing in parallel at the *subsystem layer*. The hand control subsystem performs coordinated control of finger joints, forward and inverse kinematics, acquisition of proximity, position and force sensor data, and automatic calibrations. The arm control subsystem performs predefined motion primitives, and handles serial communications between the Unival PUMA arm controller and the VMEbus control system. The vision subsystem executes a stereo vision algorithm that constantly tracks the target within the visual field. These three subsystem tasks are inherently parallel. The *system executive* is responsible for orchestrating activities among the three subsystems serially to provide fluid arm-hand motions. The *system executive* contains a set of user-defined functions that are frequently called on by CLIPS to carry out any user-defined commands. The mechanism on how these user-defined functions are integrated under CLIPS is described briefly in the Robot Programming in CLIPS section. For a more comprehensive description, one should consult the CLIPS User's Guide[4] and CLIPS Reference Manual[10].

Dexterous Arm and Hand Subsystems

The dexterous arm and hand subsystems are responsible for primitive-level control of the Utah/MIT Hand and the PUMA 560 arm. The Utah/MIT Hand is a 16 DOF dexterous hand with 3 fingers and a thumb in an anthropomorphic arrangement (see Figure 4). Embedded within each finger joint is a Hall-Effect sensor for position feedback. Each joint is controlled by two antagonistic tendons. Thirty-two Hall-Effect force sensors located in the wrist are used to detect tendon tensions. The tendons are actuated pneumatically by thirty-two air cylinders. Accompanying the Utah/MIT Hand is an analog servo controller driving the air cylinders. Position and force command signals are received from the D/A converters, and the position and force sensor signals are sent to the A/D converters. The hand controller contains several motion and sensing primitives that may be called upon by the *system executive* software running on the 386SX. The primitives running on the hand controller are listed in Table I. The *system executive* can invoke these primitives by passing command tokens, via the VMEbus, to the dual-ported memories of the hand controllers. When a primitive is invoked, the task is carried out by the hand controller SBC, and the *system executive* is now free to do other tasks. Once the primitive is fully accomplished, a DONE flag is raised to signal task completion. DONE flags are present in the hand controller, the arm controller, and the vision controller. They are extremely important for synchronization of parallel movements. Forward and inverse kinematics are both included in the hand controller software. Primitives such as TIP_MOVE, TIP_ POSITION move and report fingertip locations in Cartesian coordinates. Raw A/D counts for joint and tendon sensor readout are included to facilitate system debugging and calibration.

The arm controller operates very similar to the hand controller. The communication interface between the arm controller and the *system executive* is also through dual-ported memories. The arm controller accepts only two commands: MOVE_ARM_TIP, and MOVE_ARM_ JOINT. The first specifies positions in Cartesian space; the second specifies positions in joint space. In VAL II (a PUMA programming language) terminology, they correspond to *transformed* moves and *precision* moves. During actual operation, arm controller accepts

commands from the *system executive*, and formats these commands into proper VAL II syntax, and then sends the formatted VAL II commands to the Unival PUMA arm controller via a RS-232 serial line. A small auto-start program running on Unival controller accepts the formatted commands and moves the PUMA arm to the proper position at specified speed. Upon detection of an error, or a completed move, an appropriate code is sent back to the arm controller via serial line to signal the event.

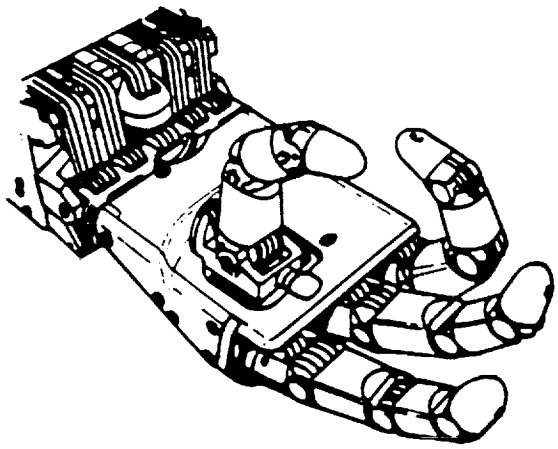


Figure 4. The Utah/MIT Dexterous Hand. (Reprint with permission from SARCOS)

Teleoperator Interface

Currently, the teleoperator interface consists of only the EXOS Dexterous Hand Master. A Polhemus 3D Tracker System, shown in Figure 5a, is being integrated with the hand master to provide a full teleoperation of the arm and hand. The EXOS Dexterous Hand Master, shown in Figure 5b, is an exoskeletal glove controller that can be worn by a human operator. The glove controller is capable of detecting movements of the first three fingers and the thumb, with four analog Hall Effect sensors per each finger and thumb. The sensor signals are amplified and filtered by a custom-built circuit board before they are passed on to the A/D converters. During operation, the *system executive* reads in the joint angles and commands the hand controller to move the robot hand to corresponding positions. The teleoperator control is presently operating in the joint space. Work is being done to include kinematic transform to allow Cartesian space control.

Vision Subsystem

The objective of the vision subsystem is to provide a 3D position of a target at a high update rate. Passive triangulation method is applied to the left and right images of the two video cameras to determine the target distance. Video cameras are used because they can capture images at a fairly high rate of 30 Hz. An active laser scanner vision system was considered. However, it was abandoned because the laser scanner proved to be slower, more costly, and less reliable. The video target tracking algorithm running on the vision controller is composed of the two modules: *centroid discovery module*, and *position calculation module*.

The *centroid discovery module* identifies the target centroid, tracks it, and performs correspondence matching between centroids found in the left and the right cameras. In order to increase search speed, the search algorithm was designed to operate in two modes. The first mode quickly searches the image by scanning from the center outward, skipping five rows at a time, alternating about the center. The search looks for pixels with a value greater than the preset threshold. Once the first of such a pixel is found, the second search mode commences at that row, working outward, comparing every row. The purpose of the search is to determine the maximum and minimum of target extent, and then take the average of the two numbers to determine the vertical coordinate of the target centroid. With the vertical coordinate of the target centroid determined, a side-to-side scanning of that row determines the horizontal coordinate of target centroid. The word *centroid* is used loosely here to denote the center of the projected target area. With the first centroid identified in one image, the second centroid can

be quickly determined from the other image. Since the cameras are co-planar, the second centroid should be located at the same row in the second image. Therefore, only a single row needs to be searched. These two centroids are assumed to be at the same point, and their pixel locations are passed on to the *position calculation module*.

The position calculation module simply calculates the target position using a triangulation method. In order to produce accurate calculations, horizontal and vertical fields-of-view (FOV) of the two cameras were measured. Knowing the FOV, the relationship between the target angle and the pixel location can be characterized by the following equations:

$$h = \frac{H}{512}, \quad v = \frac{V}{480}$$

where H and V are the horizontal and vertical FOV (in radian), respectively. The constants, 512 and 480, are the horizontal and vertical pixel image resolution of the frame grabbers. Parameters h and v are the multiplicative constants that convert pixel positions to target angles. These two equations are used in the triangulation calculations to determine x, y, and z values of the target location. To facilitate triangulation, the cameras are physically configured as illustrated in Figures 6a and 6b. The reference frame used is a right-handed coordinate system with the origin located midway between the two cameras. In this configuration, both cameras are located on the X-Y plane. The equations that determine distance in Z are:

$$Z = \frac{d\tan(\pi-\theta_1)\tan(\theta_2)}{\tan(\pi-\theta_1)-\tan(\theta_2)}; \theta_1 > \pi$$

$$Z = \frac{d\tan(\theta_2)\tan(\theta_1)}{\tan(\theta_1)+\tan(\theta_2)} \quad \theta_1 \le \pi, \theta_2 \le \pi$$

$$Z = \frac{d\tan(\pi-\theta_2)\tan(\theta_1)}{\tan(\pi-\theta_2)-\tan(\theta_1)}; \theta_2 > \pi$$

The two angles are determined by

$$\theta_1 = (\pi - hI_L)$$

$$\theta_2 = hI_R$$

TABLE I. HAND CONTROL PRIMITIVES

COMMAND	TYPE
JOINT_MOVE	JOINT SPACE POSITION CONTROL
TIP_MOVE	CARTESIAN SPACE POSITION CONTROL
JOINT_COUNT	JOINT SPACE POSITION FEEDBACK (RAW A/D COUNT)
JOINT_ANGLE	JOINT SPACE POSITION FEEDBACK
TIP_POSITION	CARTESIAN SPACE POSITION FEEDBACK
TENDON_COUNT	TENDON SPACE FORCE FEEDBACK (RAW A/D COUNT)
TENDON_TENSION	TENDON SPACE FORCE FEEDBACK
JOINT_TORQUE	JOINT SPACE TORQUE FEEDBACK

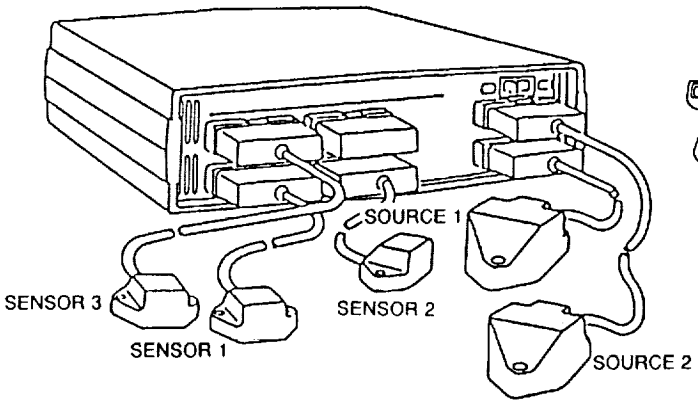


Figure 5a. Polhemus 3D Tracker System.
(Reprint with permission from Polhemus)

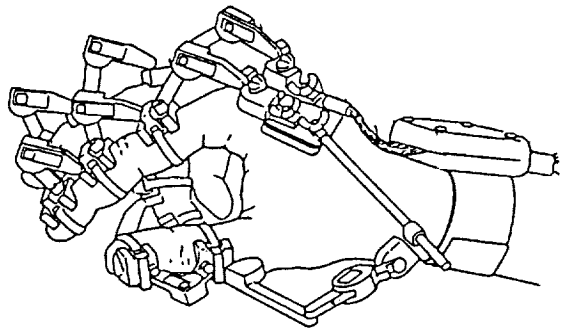


Figure 5b. EXOS Dexterous Hand Master.
(Reprint with permission from EXOS)

Variable I_R and I_L are the horizontal pixel count of target centroids in the right and left images, respectively. Once the Z component is found, values for X and Y are determined by

$$Y = Z \tan\theta_3 \quad ; \theta_3 = \mathrm{v}\, J_R$$

$$X = Z \tan\theta_4 - \frac{d}{2} \quad ; \theta_4 = h I_R$$

The variable J_R is the vertical pixel count of the target centroid in the right image.

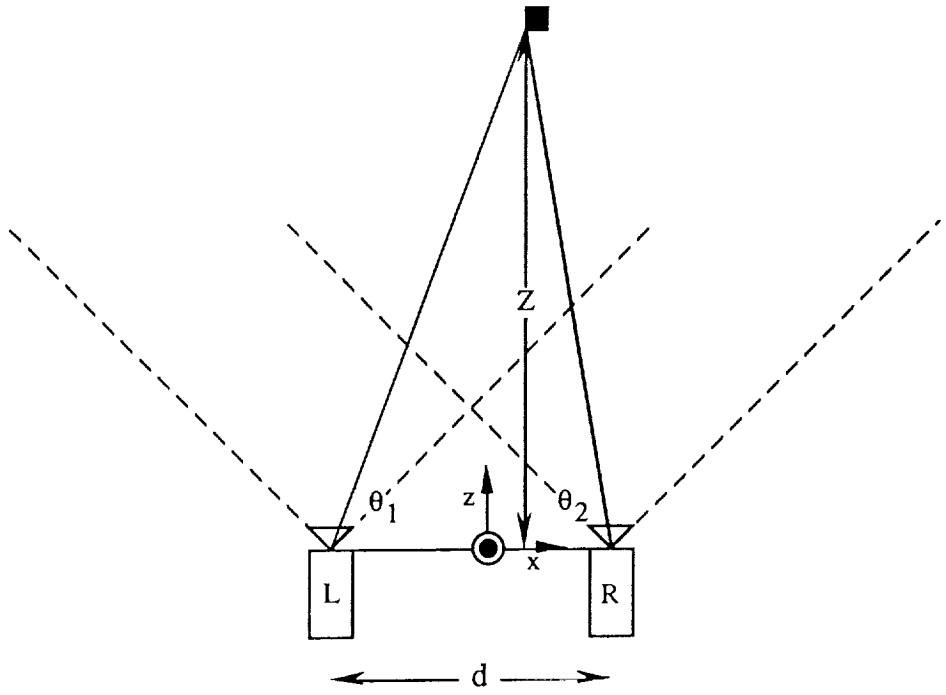


Figure 6a. Schematic of the stereo vision algorithm.

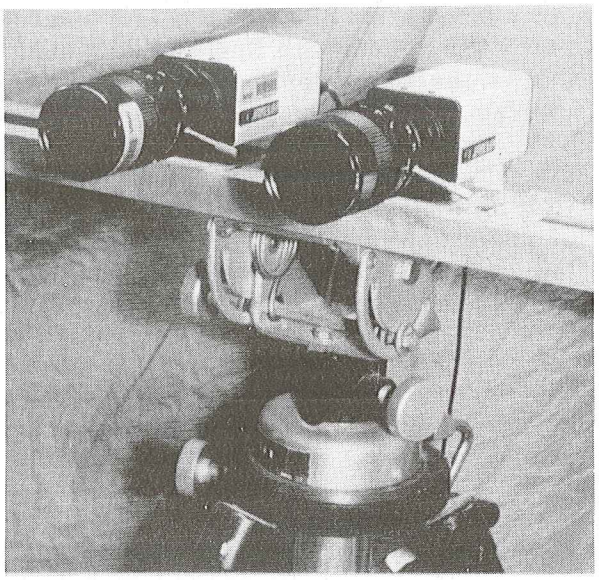


Figure 6b. Video camera configuration.

The stereo vision tracking algorithm was developed on a Sun 3/260 workstation using an Oasis C compiler. The cameras are made by Javeline, Model JE7362. The two frame grabbers are made by Data Translation, Model DT1451. The tracking algorithm, running on a 20 MHz 68020 SBC, was able to provide a target position update rate of 10 frames per second.

Proximity Sensor Subsystem

Although the vision system is capable of target tracking, it is currently not capable of dealing with visual obscurity caused by having the arm and the hand coming between the cameras and the target during an act of reaching and grasping. Without guidance from the vision system, other means of determining when to grasp (i.e., close hand) is necessary. Hess and Li[5,6] (U.S. Patent No. 4,980,626) described a method of using a proximity sensor to determine when and how to grasp a target with a dexterous end effector. Figure 7a is a drawing of the infrared proximity sensor which was originally an optoelectronics part developed by Optek, formerly a division of TRW (Part No. OPM102T). However, this part has become obsolete since then. There are eight sensors embedded inside the Utah/MIT Hand, with two in each finger, as is shown in Figure 7b. The sensors are reflective, with transmitter and receiver co-located on the same substrate. Due to its low power, the sensor does not have a very long detection range. A special signal amplifier was built to increase the signal power and to filter out noises. As a result, the sensors can now detect objects at approximately 1 to 2 inches away. The amplified signals are digitized by an A/D converter and transmitted, via a high-speed serial fiber-optic link, to the VME multiprocessing controller. No special software communication protocol was required because the fiber-optic link multiplexes and transmits signals at a high frequency of 125 MHz, thus making the interface on both sides of the link appear fully parallel.

ROBOT PROGRAMMING IN CLIPS

One of the most interesting features of the testbed system is the incorporation of an expert system shell, CLIPS, for high-level control. CLIPS has added many interesting and useful features to our system:

a. Intelligent rule-based control
b. Interactive user interface
c. User-defined functions
d. Portable C source codes
e. English-like syntax
f. Command clustering

The first feature provides machine reasoning capability for the robot. The interactive user interface allows the robot programmers to call on different robot primitives without having to recompile code every time. This feature is very important during application development and debugging. Once the entire algorithm (i.e. set of rules) is developed, it can be converted into binary form for batch mode operations. As part of the CLIPS package, a simple software interface was provided to link into user-defined subroutines. In our implementation, these subroutines provide primitive robot control functions such as MOVE-ARM or MOVE-HAND. Once the user-defined subroutines are linked into CLIPS, they become part of the

CLIPS command set. Table II lists some of the user-defined functions added to the CLIPS command set.

Source codes written in C are provided with the CLIPS software package. The codes were written in such a way that minimizes operating system and hardware dependency. The software has been successfully ported to PC, Sun 3 Unix workstations, VAX/VMS, Macintosh, and other machines. Since CLIPS is a symbolic production system with English-like syntax, it allows the robot application developer to program the robot in a descriptive language. The user-defined commands listed in Table II are some good examples. Furthermore, with CLIPS, new robot commands with a higher level of abstraction may be constructed from lower level primitives. Consider the following set of CLIPS rules:

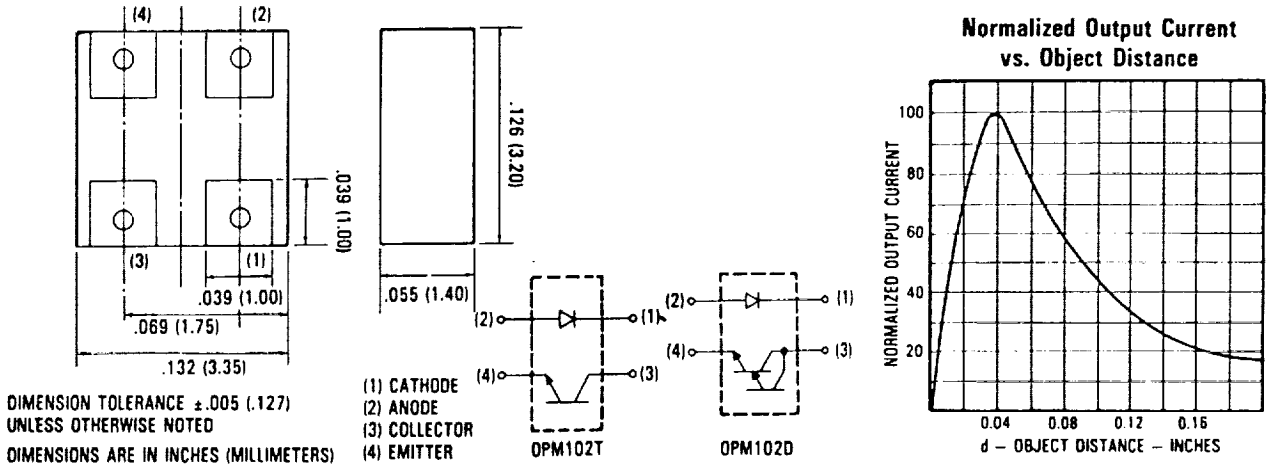


Figure 7a. Infrared proximity sensors. (Reprint with permission from Optek)

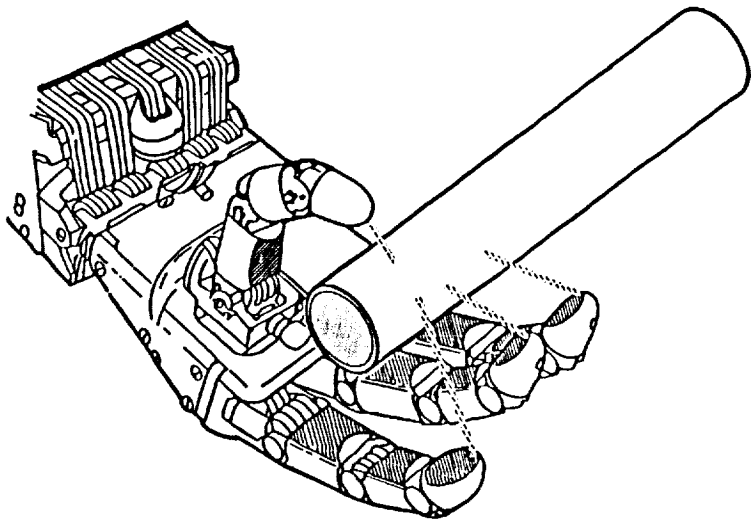


Figure 7b. Utah/MIT Hand with proximity sensors.

Rules 1, 2, and 3 are considered primitives. Rule 4 is a higher level command that produces a combined effect of having Rules 1, 2, and 3 firing together. To fire Rule 4, one simply loads in the set of rules listed above and executes the following CLIPS command:

(assert (acquire the target))

This assertion will cause Rules 1, 2, and 3 to fire, thus producing a combined action of looking, reaching, and grasping.

FUTURE WORK

The past effort has resulted in the completion of a single arm/hand system operating in both autonomous and partially teleoperated modes. The current focus is on perfecting the teleoperator system by integrating the Polhemus 3D Tracker into the system for arm control. Furthermore, the development of a dexterous robotic system with two arms and two hands is being pursued. Computer control is currently being developed for the Stanford/JPL Hand. We will be mounting the Stanford/JPL Hand on a second PUMA to form a second arm/hand system. The PC 386SX computer will be replaced by a Sun 3/160 workstation running Unix BSD 4.3. The Sun 3/160 workstation will provide a better software development environment for the programmers due to its window environment (Graphical User Interface and X Window). Neural network algorithms for learning the mapping between visual inputs and arm/hand position commands will be investigated in detail this year. Design and fabrication of a VMEbus-based neural network

TABLE II. SOME EXAMPLES OF USER-DEFINED CLIPS COMMANDS

USER-DEFINED CLIPS COMMANDS	DESCRIPTION
(move-arm *speed x y z rx ry rz*)	Move arm to the specified position and orientation with a given speed
(move-arm-home *speed*)	Move arm to predefined home position
(move-tips *speed x0 y0 z0 x1 y1 z1....x3 y3 z3*)	Move fingertips to specified Cartesian locations at the specified speed
(move-joints *speed J0 J1 J2... J15*)	Move finger joints to specified joint angles at the specified speed
(get-joint-pos) (get-visual-pos)	Get joint/target position and assert the information in the fact-list
(arm-move-done) (hand-move-done)	Returns DONE flag for arm/hand moves
(read-hand-ir)	Invoke real-time proximity sensor readout for check-out and debugging purposes
(grasp-with-ir *speed ir__threshold*)	Grasp at specified speed if infrared (IR) sensors total activity exceeds threshold
(see-reach-grasp *speed approach__distance o a t*)	Move arm/hand to visually determined target position and grasp default speed

```
; Rule 1: get target position
        (defrule  get__target__position__primitive
            ?old__fact <- (get target position)   ;  fact-list trigger pattern
        ⇒
            (retract ?old__fact)          ;delete old fact
            (get-target-pos)        ; call on user-defined subroutine to assert target position)
                                    ; information onto the fact-list

; Rule 2: arm movement primitive
        (defrule  arm__move__primitive
            ?old__fact1 <- (target position is ?x ?y ?z))   ;fact-list trigger pattern #1
            ?old__fact2 <- (move arm to target with speed ?speed)         ; fact-list trigger pattern
        ⇒
            (retract ?old__fact1 ?old__fact2)         ;delete the old facts
            (move-arm ?speed ?x ?y ?z 90 -180 0) ;call user-defined arm move subroutine)

; Rule 3: hand movement primitive
        (defrule  hand__move__primitive
            ?old__fact <- (grasp object with ir) ;fact-list trigger pattern
        ⇒
            (while (= (arm__move__done) do)  ;         wait 'til arm move is complete
            (retract ?old__fact)    ;delete the old fact
            (grasp-with-ir 1 25)   ;call  user-defined grasp subroutine for grasp)
        )

;Rule 4: acquire the target -- combining Rule 1, 2, and 3.
        (defrule  acquire__the__target__primitive
            ?old__fact <- (acquire the target)
        ⇒
            (retract ?old__fact)
            (assert (get target position))     ;this assertion causes Rule 1 to fire, producing
                                               ;(target position is x y z) fact
            (assert (move arm to target with speed 100))      ; this assertion causes
                                                              ; Rule 2 to fire, moving the arm
            (assert (grasp object with ir))   ;this assertion causes Rule 3 to fire, closing hand
        )
```

board is planned for this year. New vision hardware and software will be acquired and developed to increase the vision system performance. Finally, we will continue to investigate and to develop space robotics applications. Demonstration of candidate space-related tasks are planned for the last quarter of 1991.

CONCLUSIONS

This paper has described the implementation of a dexterous robotic testbed system developed at the NASA Johnson Space Center. The system included several desirable features found in other systems. The system is unique in the fact that all these desirable features are integrated into one system, thus making the system capable and flexible. These features include dexterous hand and arm, stereo vision, multiprocessing, rule-based programming, local hand control using proximity sensor feedback, and autonomous and teleoperator capabilities in co-existence for shared and traded controls. Some of these features were enhanced to increase system flexibility and capability. Subsystem implementations were described in detail; and examples of rule-based robot

programming in CLIPS were also given. Our experience in the past year has revealed the need for additional development in the following areas: target tracking, image understanding, dexterous manipulation, force-reflective dexterous hand/arm masters, and tactile sensing. Although more vigorous tests still need to be conducted, the initial evaluation of the system has demonstrated that dexterous robots provide more capability and flexibility than conventional robots, and therefore, have an important role in future space applications.

ACKNOWLEDGEMENT

Special thanks to Reginald Dawson and Dagoberto Rodriguez for developing the fiber-optic electronics used in our system. The assistance provided by Robert Davis, Issa Zaid, and Frank Moore is greatly appreciated. We also would like to thank Dr. Jon Erickson and Cliff Hess for reviewing the paper, and providing valuable comments and criticisms.

REFERENCES

1 Allen, P., Michelman, P., Roberts, K., "An Integrated System for Dexterous Manipulation," Department of Computer Science, Columbia University, New York, NY, 1989.

2 Allen, P., Roberts, K., "Haptic Object Recognition Using a Multi-Fingered Dexterous Hand," Department of Computer Science, Columbia University, New York, NY, 1989.

3 Clark, D., Demmel, J., Hong, J., Laferriere, G., Salkind, L., Tan, X., "Teleoperation Experiments with a Utah/MIT Hand and a VPL Data Glove," *Proceedings: NASA/JPL Space Telerobotics Conference*, Pasadena, CA, January 1989.

4 Giarratano, J., *CLIPS User's Guide, Version 4.3*, NASA/Johnson Space Center, June 1989.

5 Hess, C., Li, L., "Proximity Sensors Make Robot Dexterous," *NASA Tech Briefs*, October 1990, pp 50.

6 Hess, C., Li, L., "Smart Hands for the EVA Retriever." *Proceedings: The Third Annual Workshop on Space Operations Automation and Robotics, 1989*, NASA Reference Publication 3059, pp 441-446.

7 Jacobsen, S., C., Wood, J., E., Knutti, D., F., Biggers, K., B., "The Utah/MIT Dexterous Hand: Work in Progress," *The International Journal of Robotics Research*, Vol. 3, No. 4, Winter 1984, pp 21-50.

8 Marr, D., *Vision*, Freeman and Company, New York, NY, 1982.

9 Narasimhan, S., "Dexterous Robotic Hands: Kinematics and Control." *Master Thesis*, Department of Electrical Engineering and Computer Science, MIT, January 1988.

10 NASA/Johnson Space Center, *CLIPS Reference Manual, Version 4.3*, JSC-22948, May 1989.

11 Salisbury, K., Brock, D., O'Donnell, P., "Using an Articulated Hand to Manipulate Objects," *Proceedings of the 1987 SDF Benchmark Symposium on Robotics Research*, Santa Cruz, CA, August 1987.

12 Stansfield, S. A., "Knowledge-Based Robotic Grasping," *Proceedings: IEEE Conference on Automation and Robotics, May 1990.*

13 Stansfield, S. A., "Reasoning About Grasping," *Proceedings: 7th AAAI Conference*, St. Paul, MINN. August 1988.

14 Venkataraman, S. T., Iberall, T., (Eds), *Dexterous Robotic Hands*, Spring-Verlag, New York, NY, 1990.

15 Wolovich, W., *Robotics: Basic Analysis and Design*, HRW, New York, NY 1987.

N93-11973

Self Mobile Space Manipulator Project

H. Ben Brown, Mark B. Friedman, Yangsheng Xu, Takeo Kanade
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213

We are developing a relatively simple, modular, low mass, low cost robot for space EVA that is large enough to be independently mobile on a space station or platform exterior, yet versatile enough to accomplish many vital tasks. The robot comprises two long flexible links connected by a rotary joint, with 2-dof "wrist" joints and grippers at each end. It walks by gripping pre-positioned attachment points, such as trusswork nodes, and alternately shifting its base of support from one foot (gripper) to the other. The robot can perform useful tasks such as visual inspection, material transport, and light assembly by manipulating objects with one gripper, while stabilizing itself with the other.

At SOAR '90, we reported development of 1/3 scale robot hardware, modular trusswork to serve as a locomotion substrate, and a gravity compensation system to allow laboratory tests of locomotion strategies on the horizontal face of the trusswork. In this paper, we report on project progress including the development of:

- adaptive control for automatic adjustment to loads
- enhanced manipulation capabilities
- machine vision, including the use of neural nets, to guide autonomous locomotion
- locomotion between orthogonal trusswork faces
- improved facilities for gravity compensation and telerobotic control.

Session A6: SYSTEM PLANNING

Session Chair: Jack Pennington

N93-11974

Automation and Robotics for COLUMBUS

An Implementation Concept for the Free Flying Laboratory (MTFF)

G. Götz, B. Sommer

German Space Agency, DARA GmbH, D-5300 Bonn

Abstract

With nearly fourty percent of the funding, Germany is the main contributor to the European COLUMBUS Programme, followed by Italy, France and further ESA member states. The COLUMBUS elements are the Attached Laboratory (APM) to be permanently attached to the Space Station FREEDOM, the polar platform (PPF) and the Man Tended Free Flyer (MTFF). The latter element is regarded to be of special interest for the german micro-g community.

Until now the implementation of A&R Technologies has not been included as part of the system concept for the COLUMBUS laboratory modules. Yet especially for the Free Flyer a high degree of A&R will be indispensible.

An A&R system concept and implementation options for A&R are given to make the COLUMBUS labs "intelligent" laboratories in orbit.

1. Introduction

1.1 The COLUMBUS Elements

The space segment of the COLUMBUS Programme [1,2,3] - the european contribution to the International Space Station FREEDOM (ISS) - consists of the three elements

- Attached Laboratory (APM);
- Polar Platform (PPF);
- Free Flying Laboratory (MTFF).

The APM will be permanently attached to the ISS. Its primary utilisation will be research in the fields of life sciences and materials sciences under the conditions of microgravity. The launch date for the APM, currently proposed for June 1998, and will depend on the assembly sequence of the ISS.

The PPF will be a complementary element to the U.S. polar platform. According to the present plan the platform could be ready for launch in 1997. To allow sufficient time for the development of the PPF P/Ls, the launch might be delayed for a year.

The MTFF in its baseline design consits of a pressurized laboratory (PM) and a resource module (RM) attached to the lab. Its orbit is planned to boomerang the ISS and be serviced there, which means that the MTFF had to return to the space station and be berthed and docked. A second servicing option for the MTFF is that it will be visited by HERMES, the latter launched by ARIANE 5, for a period of some days twice a year (latest planning: once a year). The manned HERMES spacecraft and the launcher AR5 are both currently under development in resepctive european programmes which are managed - as the COLUMBUS programme is - by the European Space Agency ESA.

Following latest considerations in the programme [4] the APM might be reduced in size. This would make it possible to launch the APM fully equipped with its P/L and takes into account that during the initial phase the ISS will be operated in a man tended mode until it reaches the "permanently manned capability" or even its "eight man-crew capability".

Latest plans for the MTFF propose to make it completely independant from the space station. This would simplify the interfaces as the MTFF would then only need to be compatible with HERMES. The RvD capability of the MTFF - that means the design of the RM - could be significantly reduced as HERMES in that case will be the chaser and MTFF the target only. The operational life of the MTFF would be limited by the life of the solar panels. The replacement of those is beyond the planned capabilities of HERMES.

1.2 Operational Aspects

The idea of the Space Station FREEDOM programme and consequently the COLUMBUS programme - as it started in the middle of the last decade - was to establish a permanently manned infrastructure in a low earth orbit. The planned resources in orbit, the in orbit servicing capability and the continuous schedule of access to space were to fulfill the requirements of potential users to a near optimum. A crew of eight seemed sufficient for the operation of the space segment.

In Europe the operational envelope of the MTFF - man tended pressurized laboratory, visited twice (once) a year for some days by a crew of three for servicing and optional manned experimentation - seemed to demand a high degree of internal automation and robotics for its proper nominal operation.

A more thorough analysis of the time required for P/L operations in the APM (equipped with a set of model P/Ls, based on SPACELAB experience) versus the available crew time, derived from a possible crew of eight and the terms given in the MOU [5] between the participating partners in the space station programme, revealed that crew time would not be sufficient [6]. It clearly indicated that the APM will need a relatively high degree of automation and robotics in order to make the best use of it in an scientific and economical sense.

In Germany, partner in the COLUMBUS programme sharing 38 percent of its cost, the lack of automation and robotics technology in the COLUMBUS baseline design was estimated as a severe deficit concerning the utilization and operability of the elements of the programme. Consequently two activities were started. First, in order to demonstrate the available technology during the '93 SPACELAB D-2 mission the project "Robotic Technology Experiment" (ROTEX) [7] was initialised. Currently ROTEX has passed its system test and expects its integration into the SPACELAB module. Second, a serious of studies was launched on basis of a national A&R promotion concept [8]. Among those the study "Modules as Intelligent Laboratories in Orbit" (MILORT) [9] which worked out an A&R concept and a preferred implementation option for the planned laboratories in orbit, taking the free flying laboratory as the most challenging example.

2. Definition of the A&R System Concept

The definition of the A&R system could take into account the results of a number of related studies and projects, e.g. EMATS [10], which had already been performed or which were running in parallel under ESA or national contract.

2.1 The MTFF Configuration

The technical concept of the MTFF which was taken as baseline for the A&R system concept was taken from the COLUMBUS C/D proposal delivered to ESA in 1989.

The pressurized laboratory module of the MTFF forms a shell with an inner diameter of 4216 mm and accomodates 4 rows of 8 single rack equivalents (SRE) according to the "International Standard Rack". The inner arrangement is "1-g" oriented with a floor and a ceiling. It houses 4 stand-offs in symmetrical configuration to accommodate the 4 rows of racks.

A free area of approximately 2197 mm * 2197 mm between the racks gives working space for the crew. Protrusions of up to 100 mm are allowed at any rack location.

Utility lines are connected to the stand offs and routed through the rack bottom structure to the rear side. The necessary P/L cooling is accomplished by the avionics air cooling loop and/or by the water cooling loop. Each of these loops is designed for a maximum total heat load of 4 kW.

The EPS provides 4.8 kW peak / 4.0 kW average to payloads. For each lateral P/L double rack location the available power is 3 kW peak / 2 kW average. Voltage level at rack I/F is 120 V DC.

The communication subsystem provides a Ka-band downlink with 100 Mbps and a 2.5 Mbps uplink for transmission of multiplexed system and payload data. Up to 6 P/L data medium rate channels can be transmitted simultaneously to ground. A 5 minutes gap in transmission is allowed for turning the antenna from one to the other data relay satellite (DRS). The video interface will be standard RGB plus synchro signal. This allows either one color signal or three monochrome signals to be transmitted simultaneously. Two P/L video channels are available. Tape recording is available on-board.

P/L services include the P/L manager unit, LAN communication, central database service and storage, telemetry/telecommand services, data acquisition and command via standard transmission and acquisition units (STAUs) and monitoring/limit checking by the P/L manager unit. A caution and warning system will be available.

2.2 Tasks for Internal Automation and Robotics

The following list gives an overview on the possible tasks which will have to be performed by the internal A&R system:

Payloads

- nominal payload handling;
- servicing, includes inspection, cleaning, replacement of consumables, retrieval and disposition of products and waste;
- modification, reconfiguration.

System

- nominal operation;
- servicing, includes inspection, cleaning, replenishment of consumables, preventive and corrective maintenance and implementation of upgraded equipment.

2.3 A&R System

The A&R system will consist of several parts forming a hybrid system (fig. 1). The system as a whole might be described hirarchially by 3 shells, the system shell, in which the facility shell is embedded which itself contains the experiment shell.

2.3.1 Robotics

A robotic subsystem consisting of two 6-DOF manipulators each based on a 2-axes gantry which allows vertical and horizontal movement through the laboratory will perform the

tasks involving manipulation or inspection, such as sample exchange or else. It will be operated in different modes:

- from ground in teleoperation / telemanipulation *)
- from ground via predefined path planning *)
- from ground via command macros
- o/b executing predefined tasks *)
- o/b autonomous

*) will be demonstrated by ROTEX on SPACELAB D-2 mission

The horizontal rails (HRU) of the both sides of the laboratory are interconnected in the aft cone. This enables each robot to move to the opposite side of the lab (fig. 2).

The workspace of one of the manipulators is determined by its kinematic parameters:

- longitudinal range (rail system)	5433 mm
- vertical range (rail system)	1655 mm
- streched arm	1430 mm
- shoulder joint	360 deg.
- elbow pitch	240 deg.
- 1st and last wrist	360 deg.
- middle wrist (axisis no. 5)	240 deg.

It can be seen that the the lab will be equipped with a manipulator subsystem of high dexterity.

The architecture of the robot control system (fig. 3) will follow the NASREM [11] approach. The DARA/ESA project MARS/ARCOS which is about to start will define and implement such a robotic control system for A&R ground testbeds at ESA/ESTEC, Noordwijk (Netherlands) and in Germany. The implementation will be the basis for the development of the flight system.

The robotic subsystem includes sensors which either are manipulator internal or external. The sensor concept comprises

- angular encoders (17 bit resolution)
- linear encoders
- force/torque sensors (ranges: force 10N, torque 20 Nm)
- distance sensors
- optical sensors (vision system)

The proposed End Effector (EE) concept foresees an interchangeable standard EE. This will allow to attach special tools directly to the manipulator arm.

2.3.2 Payload Internal Automation

It is assumed that the scientists will be responible for the P/L facilities and the experiments to be performed. It is also assumed that the responsibles will make extended use of hardautomation in order to reach a high performance. From a system point of view, only the A&R services and interfaces need to be defined and a set of standardized building blocks might be offered:

The electrical components relevant to automation are

- actuators, switches, relays, sensors, transducers, transmitters, pick-ups, motors.

More complex modules are

- single board and/or modular controllers, micro-computers, intelligent driver/controller modules.

In the frame of the COLUMBUS programme basic modules (CPU, memory, network interface unit (NIU), ...) are already under evaluation. Additional cards with special I/O or intelligent co-processors will have to be developed.

A standardized S/W architecture should be used which will be identical for MTFF subsystems and A&R. It will run on all nodes of the avionics H/W architecture consisting of 4 major layers: - the network layer, the operating system layer, the servicing layer and the application layer.

2.3.3 Expert Systems, FDIR, Mission Planning

The investigation of the applicability of expert systems and artificial intelligence to robotics on-board the MTFF has been guided by pragmatic considerations. According to this

- robot FDIR (fault detection, isolation and recovery)
- integrated mission and robot activity planning and execution

were identified as definitely required for the realization of on-board robotics.

The main purpose of robot FDIR will be

- to ensure safe operations
- to maximize mission success
- to provide exact information for servicing and repair to be performed during the rare visits by man.

The FDIR system foresees component failure handling (CFH) and task failure handling (TFH). As TFH is much more complex than CFH, it is expected that the initial on-board capability will provide only for the detection of deviations and corresponding saving actions. Diagnosis and repair as well as replanning shall be performed on-ground.

On-board operations of sub-systems and payloads will be defined by the Master Timeline and executed by the system and mission management of the MTFF. To include a robotic system into on-board operations will mean to

- include robot actions into the Master Timeline generation,
- include robot-specific task decompositions - and its complement on the P/L side - for o/b execution into the sytem and mission management plan.

2.3.4 Man Machine Interface

The traditional MMI consisting of video screens, switches and joysticks may not be advanced enough for full operation of the MTFF from ground. The ROTEX D-2 ground station will use in telemanipulation mode video screens, a predictive graphics system and sensor-balls to control the robot on-board SPACELAB D-2. There will be no force reflection to the operator.

A more advanced MMI could possibly be reached making use of techniques which today are summarized under the name of "virtual reality". Extensive use of such technologies could give a new quality to future manned space flight: to be virtually in space. That is to feel, to see, to work, to do experiments as if one would be in an orbital laboratory, in an observatory on the rear side of moon or at some other location in space, but physically to be on earth.

DARA will kick-off a project named VITAL to demonstrate and assess the possible implementation of "virtual reality" techniques into the man machine interface of the MTFF's automation and robotic system.

3. Implementation Options

The implementation of a complex system as the automation and robotics system into the COLUMBUS programme might be performed in different ways. The A&R system may not be considered as purely another subsystem or a user facility like the other experiment set-ups. In principle three different implementation options can be identified, implementation as:

- (1) autonomous payload
- (2) payload supplied facility
- (3) subsystem

For each of the above mentioned A&R elements an individual implementation strategy might be given.

- central robot system	>	(?) different options avail.
- payload internal automation	>	(2) integral part of P/L
- robotic FDIR system	>	(3) extension of nominal system
- robotics mission management	>	(3) integrated in nominal system
- man machine interface	>	ground segment

Two trades were performed for the central robotic system in order to find the optimal concept. As a constraint applicable to both trades it had to be regarded that the COLUMBUS programme was in its transition from phase B into the development and construction phase C/D.

Trade 1, technical criteria:

- reliability, availability
- safety
- materials selection
- COLUMBUS standards
- impacts on space segment design
- impacts on space segment operations
- ease of implementation
- ease of adaptation
- concept definition
- concept analysis
- concept trade evaluations
- number of interfaces
- complexity of interfaces
- manufacturing
- S/W development

Trade 2, programmatical criteria

- development control
- documentation effort
- development risks
- integration steps
- implementation schedule
- verification criteria definition
- verification method evaluation
- verification level definition
- integration sequences
- integration schedule
- add-on development responsibility
- administration for operations

The result was that the subsystem option was seen as the preferred option for the implementation of the central robotic system (fig. 4,5,6).

The development of the A&R system (fig. 7) depends on several support facilities. Major support functions are performed by testbeds where the A&R technology and operations may be developped, verified, analyzed, demonstrated and improved. A number of testbeds are under construction in Europe, some of those in a quasi-finished operational status.

MARS - Modules A&R System Testbed	Bremen (Germany)
STA - Simulation and Test Assembly	Oberpfaffenhofen (Germany)
CAT - COLUMBUS Automation Testbed	Noordwijk (Netherlands)
CWST - Crew Workstation Testbed	Noordwijk (Netherlands)
TTB - Telescience Testbed	Noordwijk (Netherlands)

For the in-flight demonstration of A&R technology in preparation for COLUMBUS the ESA "COLUMBUS Precursor Flights" programme [12] will be used. One of the A&R oriented proposals is "BIOSCREENING". The purpose of this experiment - seen from an A&R point of view - is to demonstrate the co-operation of a robotic arm and the Biorack facility. The experiment should fly on the European SPACELAB E-1 mission in '94.

4. Conclusions

The need for an A&R system in the COLUMBUS laboratories has been proved by several studies and is generally accepted now. It has been shown that although the programme is moving in the development and construction phase it is not too late to bring in an A&R system, partly as an additional subsystem partly as payload supplied facility. The results for the free flying laboratory in principle can be applied also to the attached lab.

It now needs some effort by the partners in the programme COLUMBUS to bring the development of the A&R system under control of the COLUMBUS system development at ESA. When in the past ESA followed the "permanent manned" option it now seems that there are strong hints that ESA will change its attitude. J.M. Luton, the Director General of ESA, stated recently [13]: "... Astronauts operating manned systems should be used only for essential tasks which cannot be performed without human intervention..." and "... The free-flyer, which can be visited and tended by astronauts from time to time with the help of HERMES, sets the proper course for the future development of robotics."

[1] "Resolution on Participation in the Space Station Programme", 31-Jan-85, ESA/C-M/LXVII-Res.2 (final)

[2] "Resolution on the European Long Term Plan and Programme", 10-Nov-87, ESA/C-M/LXXX-Res.1 (final)

[3] "Declaration on the COLUMBUS Development Programme", 15-Dec-87, 29-Jun-89, ESA/PB-COL/XVIII-Dec.1 (final), Rev.3

[4] ESA/C-WG(91)WP/22, 14-May-91

[5] "MoU - Übereinkünfte betreffend die Raumstation", 18-Oct-88, ESA/C(88)74

[6] "ROSSA - Robotics Spacecraft Servicing and Assembly in Space", ESA Contract 6837

[7] "ROTEX - Robotic Technology Experiment on Spacelab D-2", DARA Contract 01TT8701

[8] "Automatisierungstechnologien für die Raumfahrt", Bonn 15-Mar-89, BMFT

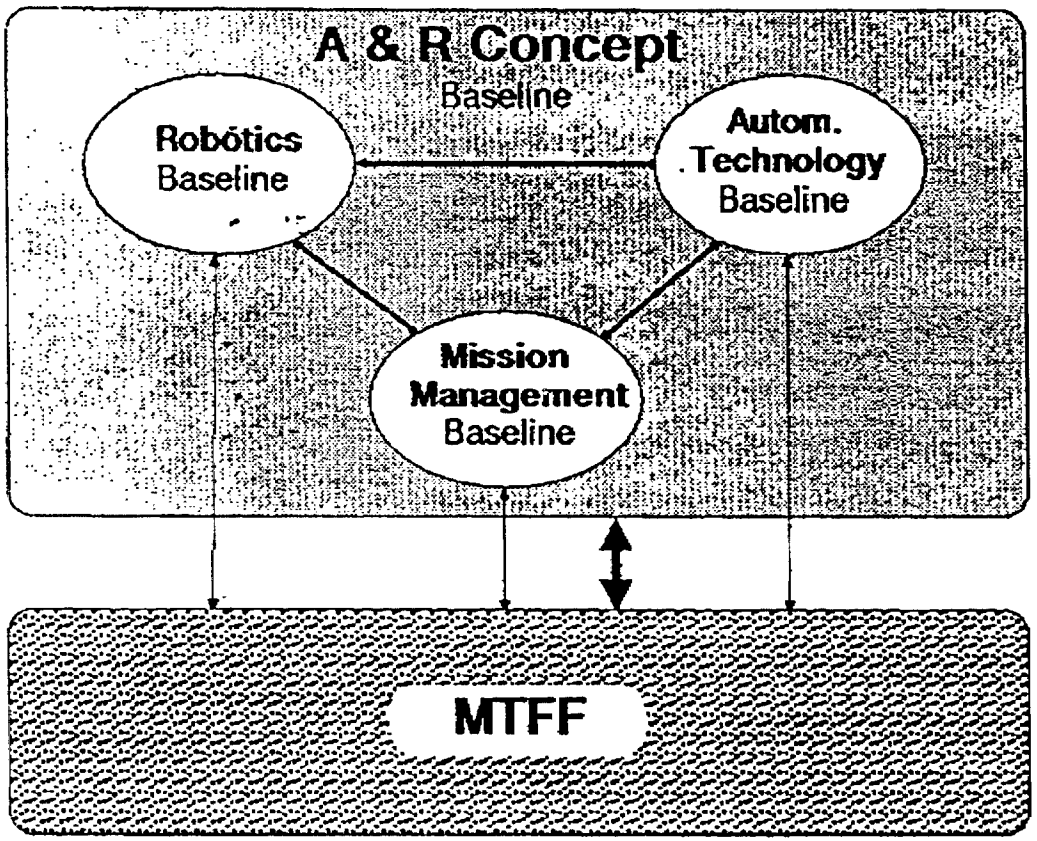
[9] "MILORT - Modules as Intelligent Laboratories in Orbit", Final Report, Bremen 1-Aug-90, DARA Contract 01 RS 8855

[10] "EMATS - Equipment Manipulation and Transportation System", Final Report, ESA Contract 7412/87

[11] AIAA-87-1687 "Software Architecture For Manufacturing And Space Robotics", J.S.Albus et al., 2nd AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program, Arlington VA 9/11-Mar-87

[12] "COLUMBUS Precursor Flights", Call for Proposals and Ideas, ESA Paris 1-Nov-90

[13] ESA Bulletin No. 66, p. 9ff., 1-May-91, Paris ESA/HQ



Robotics
- Central Robot System

Mission Management
- Robotics Mission Management
- Failure Detection, Isolation, Recovery (FDIR)

Automation Technology
- Communication/DMS
- BITE
- Payload Internal Automation

Mission Planning

COLUMBUS FREE FLYING LABORATORY (MTFF)
- Interfaces
- Architecture
- Operational concept

Fig. 1 A&R Baseline Concept

Central Robotics System (EMATS)

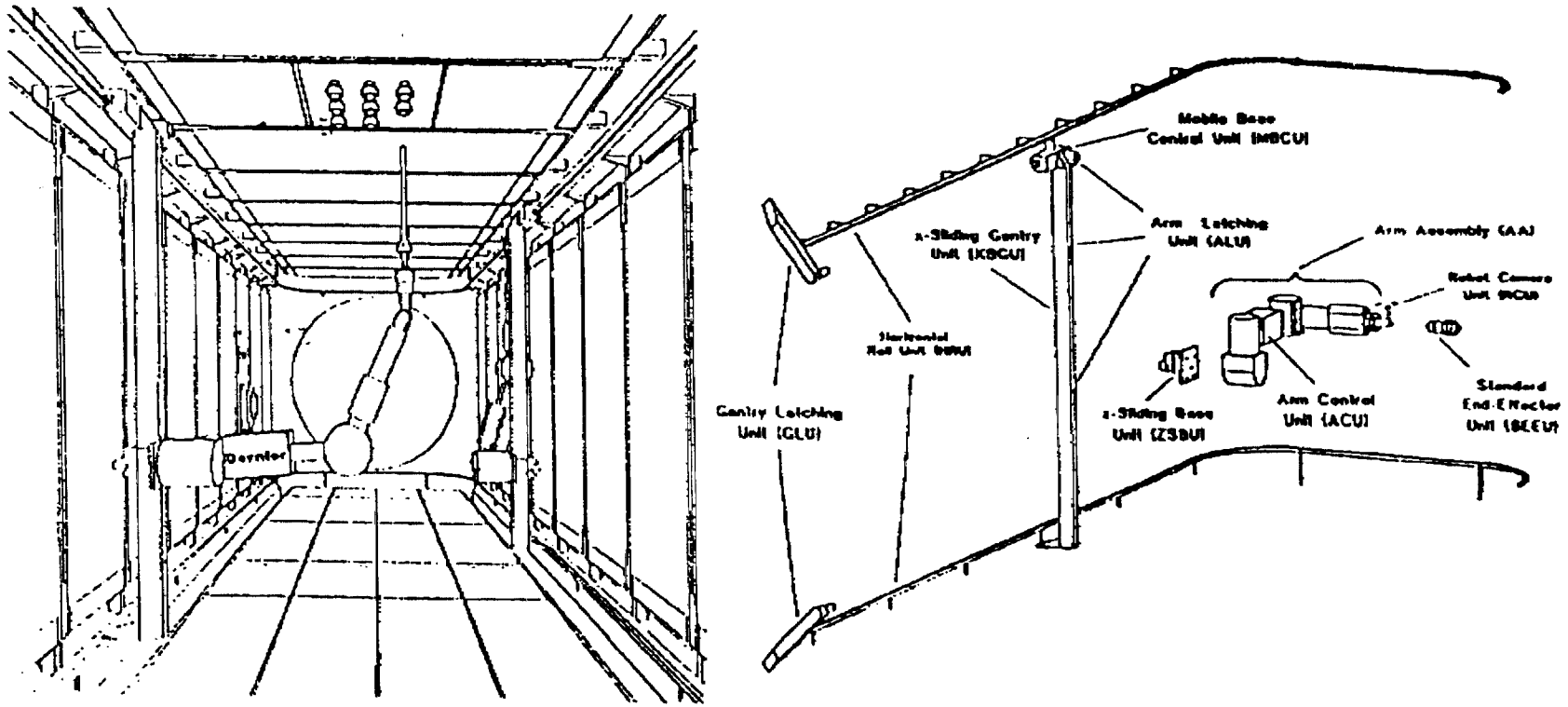


Fig. 2 Central Robotics System

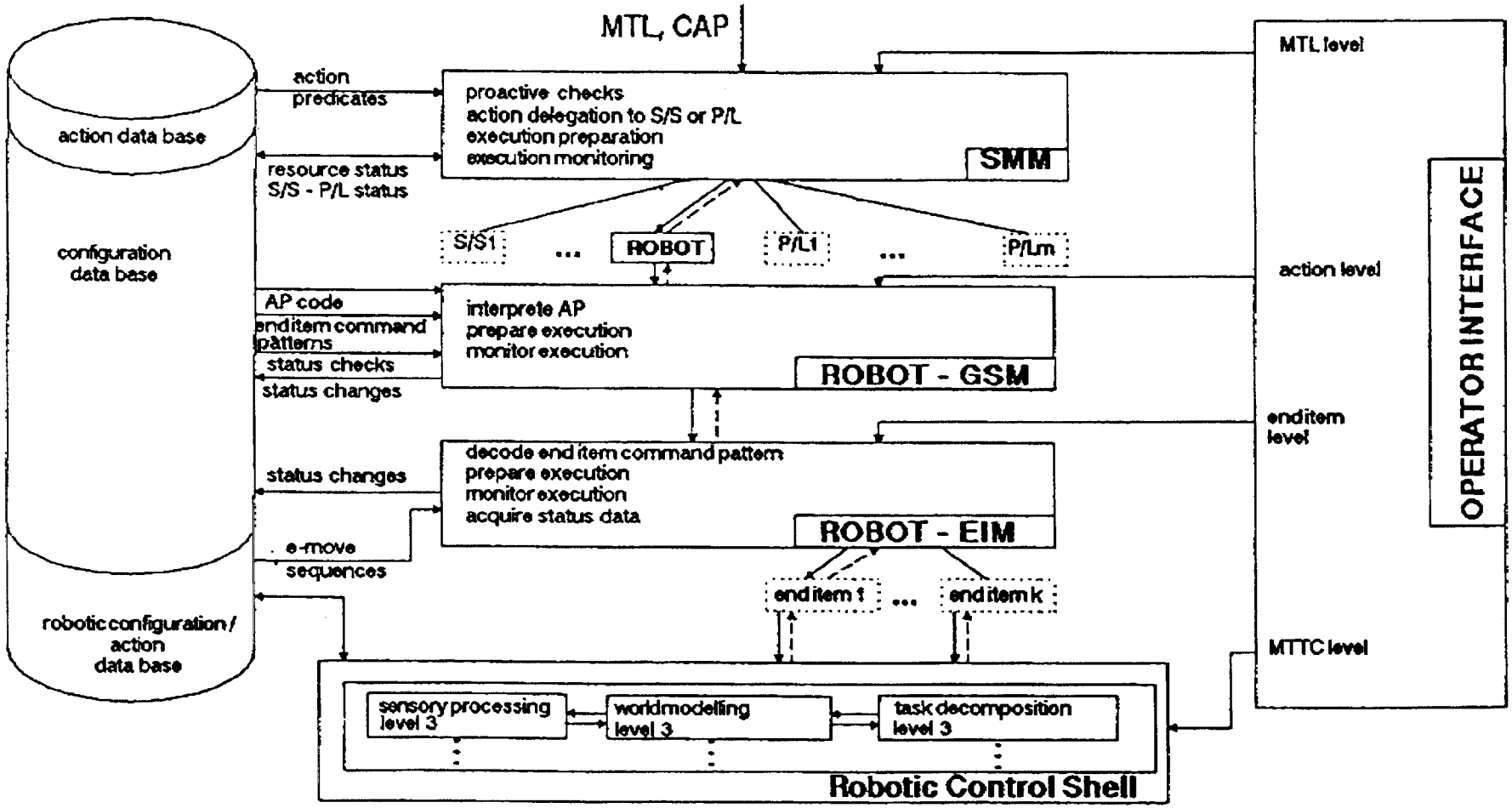


Fig. 3 Robotic Control System - Architecture

For the implementation of central robotics syst in the MTFF (APM) two different options are availal

Implementation of central robotics system as

PAYLOAD SUPPLIED FACILITY

Implementat of central robotics syste

INTEGF ED MTFF S=SYSTEM

The implemcatation options differ in:

Technical impacts — Degree of Integration into COLUMBUS Sys evelopment

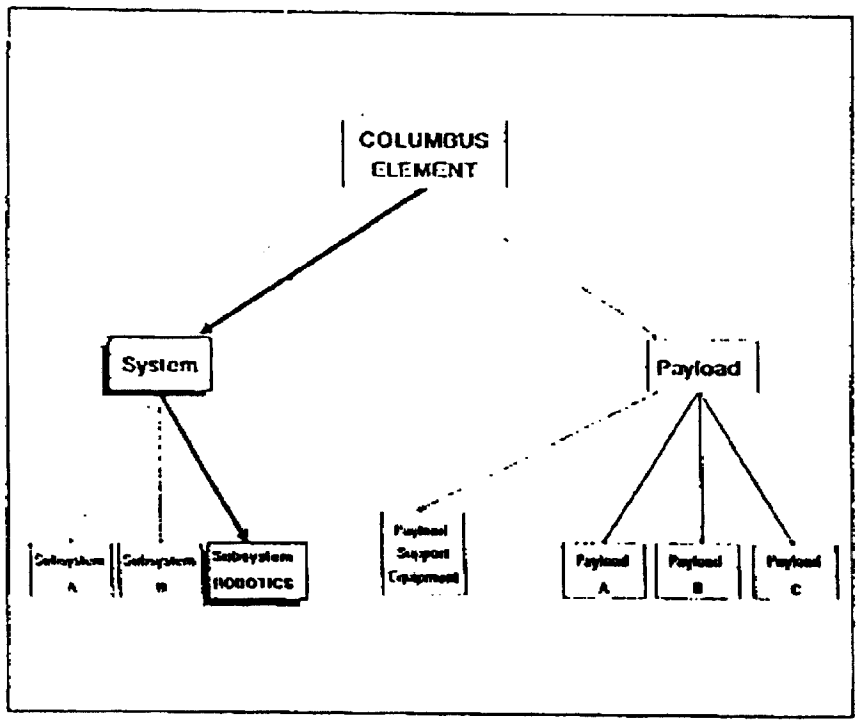
- Technical Requirements
- System Engineering
- Interface Definition/Management
- Qualification Standards, AIV

Programmatical impacts — Degree of Integration into COLUMBUS Fro Development Plan

- Independent Project vs. integral p art of C BUS Program
- Allocation of Budgets (mass, power, volu c.)
- Financial Resources

Fig. 4 Implementation Options (1)

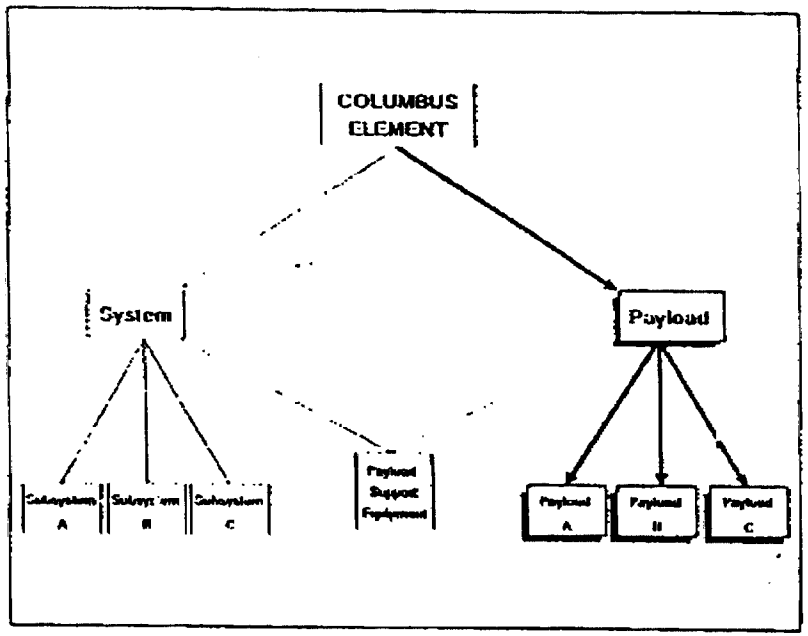
Implementation of Central Robot as COLUMBUS Subsystem



- **Optimization of on-board overall System for MTFF and APM**
- **Commonality between MTFF and APM**
- **Integral part of COLUMBUS Program Development Plan**
- **Extension of COLUMBUS C/D contract for Robotics subsystem development**
  - allocation of additional budgets
  - allocation of additional financial resources
- **Possibility of step-by-step integration of Robotics subsysytem**
  - rails prior to launch
  - integration of 1. Robot prior to launch preferred
  - integration of 2. Robot until 3. Servicing cycle recommended

Fig. 5 Implementation Options (2)

Implementation of P/L Internal Automation as Payload Supplied Equipment



- **Implementation of P/L Internal Automation as integral part of the payload**
  - Multi User Facilities
  - Payload Instruments

- **Establishing of a set of fully developed and qualified standard automation equipment to enable an optimum and costeffective payload development.**

Fig. 6 Implementation Options (3)

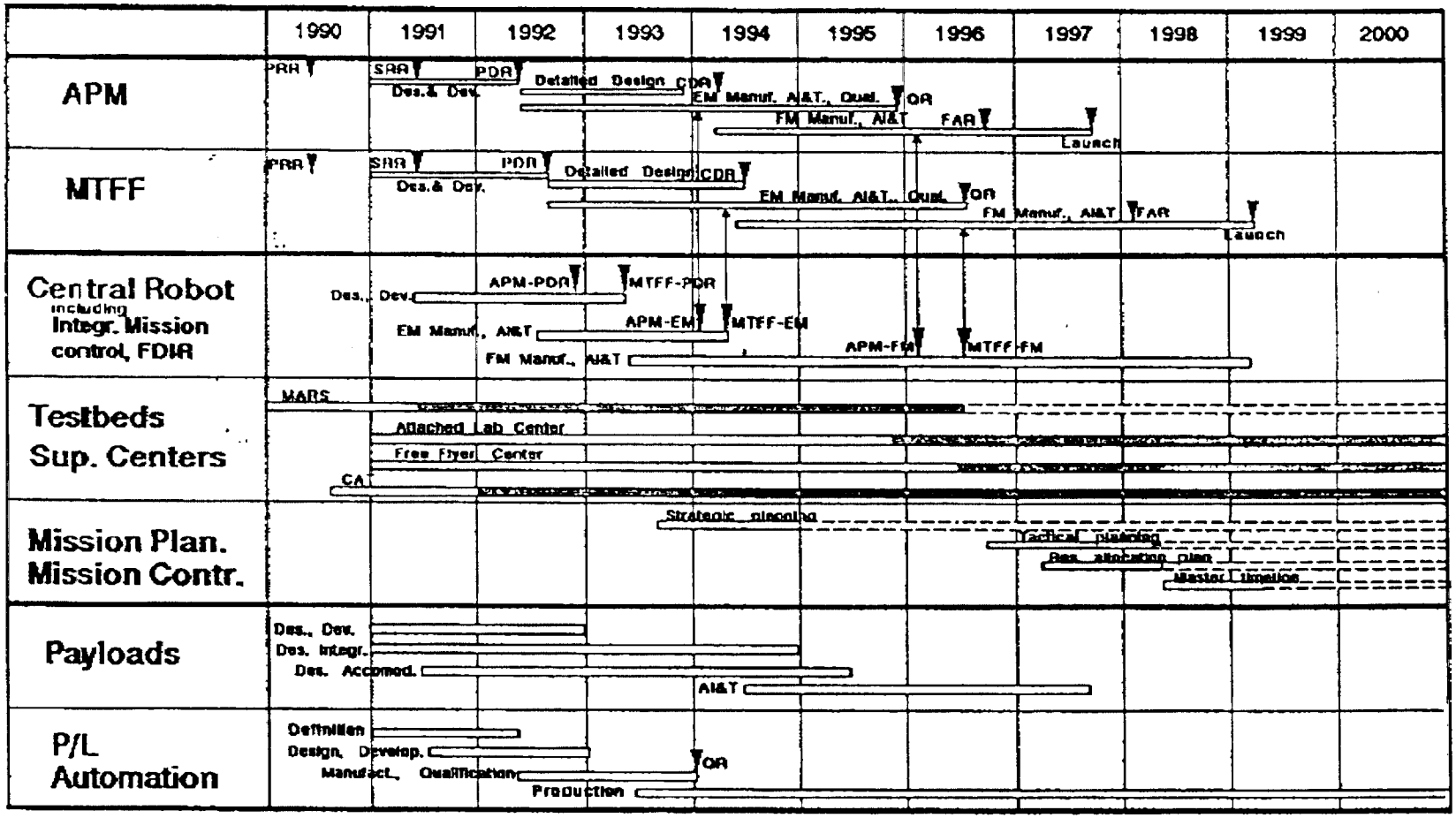


Fig. 7 A&R Development Plan

N93-11975

SPACE STATION ROBOTICS PLANNING TOOLS

Bridget Mintz Testa/Barrios Technology
Mission Operations Directorate/JSC/DF44
Space Operations, Applications, And Research Symposium
July 9-11, 1991

1.0 Introduction

Planning for a complex Shuttle Remote Manipulator System (SRMS) mission can require close to 5,000 manhours. This time is used for kinematic, dynamic, and clearance analyses, procedures development, verification, validation, and construction of the Flight Data File. SRMS missions, with their one-shot, one-robot nature, have been able to accommodate 5,000-manhour planning schedules. The Space Station Freedom Program (SSFP) cannot accommodate manhour-intensive planning schedules for multiple, international robots over a 30-year lifetime of changing configuration and continuous operations.

In this paper, we will describe the concepts for the set of advanced Space Station Freedom (SSF) robotics planning tools for use in the Space Station Control Center (SSCC). We will also show how planning for SSF robotics operations is an international process, and we will indicate baseline concepts for that process. Current SRMS methods provide the backdrop for this SSF theater of multiple robots, long operating time-span, advanced tools, and international cooperation.

2.0 Present RMS Planning Tools and Flight Data File Development

SRMS planning begins several years prior to the launch of the payload, with a study of the payload to be deployed, retrieved, or manipulated. These activities examine the payload's compatibility with the Space Shuttle and ensure that the payload customer's requirements are met. Examples of payload customer requirements include thermal, pointing, or plume-impingement constraints.

Performance of these initial payload assessments includes identification of the required analyses. Initial analysis is nearly always kinematic; the primary SRMS kinematic analysis tool is the RMS Planning System (RPS). The second step is a dynamic analysis using the Payload Deployment and Retrieval System Simulator (PDRSS). The third step is detailed clearance analysis using the Clearance Analysis Tool (CAT). Other simulators will be used to verify and validate procedures. The final outcome of the analysis process is the Flight Data File (FDF), which is the set of procedures astronauts follow onboard to operate the SRMS. This entire process is highly iterative.

Each of the SRMS analysis and procedures development tools is independent. However, results of one tool often affect the initial settings for another simulator. Subsequently, data sets are exchanged.

2.1 RPS Tool

RPS analysis (including model development) for a complex flight averages 520 manhours,

beginning two years prior to the launch of a payload (L-2). Analyses performed on this 3-D graphics kinematic simulator include:

- payload grapple fixture location
- trajectory definition
- reach, visibility, and approximate clearance
- initial procedures and timelining
- initial flight software parameter definition (Level C data tapes).

Until this year, RPS was hosted on an HP9000™ platform. It has now been re-hosted to a Silicon Graphics Iris™ workstation.

2.2 PDRSS Tool

PDRSS performs non-graphical dynamic analysis for SRMS. Formerly hosted on a Univac™, PDRSS has now been re-hosted to Silicon Graphics Iris™ workstations. Re-hosting the simulator to these workstations enables output of PDRSS data files to Clearance Analysis Tool (CAT) in "playback" mode to drive a graphical simulator. Thus, results of dynamic analyses can now be viewed.

Examples of PDRSS analyses include:

- capture loads
- appendage jettison
- loads
- RCS-induced loads and motion
- maneuvering loads
- other externally applied loads

PDRSS math models include:

- servo-mechanisms
- rigid and simplified flex SRMS models
- latest flight software for modelling RCS forces and moments
- orbital mechanics models.

The Shuttle program formerly performed extensive analyses to determine impacts of an SRMS runaway. However, SRMS flight operations combined with reliability analyses of runaways have shown that the probability of an SRMS runaway is extremely low. Therefore, in March, the Shuttle program eliminated requirements for runaway analyses.

Two additional dynamic simulators are used for RMS planning:

- Draper Laboratories' Draper RMS Simulator (DRS)
- Spar Aerospace's All Singing, All Dancing (ASAD).

DRS is used primarily to analyze Digital Auto Pilot (DAP) stability and DAP initial software loads (I-loads). ASAD has full payload-flex models.

2.3 CAT

The CAT platform is a Silicon Graphics Iris™. The software used is Deneb IGRIP™, which is a 3-D graphics commercial software. Data files of properly sequenced joint angles for an SRMS trajectory are delivered to CAT. These data files are then used to drive the CAT graphical simulator to determine detailed clearances and visibility.

For a complex SRMS payload, CAT/PDRSS analyses begin about 18 months prior to the launch. The analyses (and model development) require full-time effort for that period, for a total of 2880 manhours.

2.4 Shuttle Engineering Simulator (SES)

Much of the analysis and basic procedure outlines required for a mission are completed one year before the flight. At that point, mature procedure development and verification begin, using the Shuttle Engineering Simulator (SES). Timelining of tasks and approximate clearance assessments will also be performed in the SES. SES time per payload averages 112 hours.

2.5 Shuttle Mission Simulator (SMS)

The primary use of the SMS is for training flight crew and flight controllers. However, SRMS planners also use SMS to validate procedures. These sessions represent about 50 additional manhours per year per payload, if the validation sessions can "piggyback" on the training sessions. Additional sessions are very possible.

Procedures are validated in the Shuttle Mission Simulator (SMS) at a cost of about eight percent of the total procedures development time or about 50 manhours per year. *This is RMS Section time*; it does not include training personnel time, crew time, SMS personnel support time, etc.

2.6 Flight Data File Development

Flight Data File (FDF) preliminary development begins 10 months prior to a mission. Basic procedures are published 3.5 months prior to a mission, and final procedures are published one month prior to a mission. Because the FDF development is based upon earlier analysis and may occur during the analysis period, an exact assessment of required time is difficult. However, approximately half a man-year, or 1040 man-hours, is estimated for this aspect of planning.

To estimate the actual manhours required for planning RMS missions, we can sum the estimates for the various types of analysis and planning. This exercise is shown in Table 1-RMS Planning Manhour Requirements.

Table 1-RMS Planning Manhour Requirements

Type of Activity	Manhours Required
RPS	520
Dynamic/Clearance Analysis	2880
Procedures Development/Validation	180
FDF	1040
TOTAL	4620

3.0 SSCC Robotics Planning Tools

As Table 1 indicates, the burden of analysis/planning activities for Shuttle arm mission designers and analysts is very heavy. As Section 1.0 stated, SSF cannot accommodate this burden. Subseqently, requirements have been written for a set of SSF robotics planning tools which will reduce the burden. Those requirements define a set of integrated tools for use in the SSCC which perform kinematic, clearance, and dynamic analysis, and which also develop preliminary procedures. These planning activities will be performed by an integrated team of US/Canadian mission controllers for US/Canadian onboard robots. Although Japan will have the Japanese Experimental Module Remote Manipulator System (JEMRMS), Japanese personnel will be solely responsible for analysis and procedures development for their system.

The entire environment for planning/analysis/procedures development is called the Robotics Task Analysis Environment (RTAE). RTAE also incorporates the Robotics Task Library (RTL),

and it will utilize the Graphical And Mass Properties (GRAMPS) Library. Plans are to host these tools upon a platform compatible with the Silicon Graphics™ platforms used by Shuttle arm planners to insure continuity/cooperation with them. All tools will be highly graphical, interactive, and integrated.

3.1 Robotics Task Analysis Environment

The primary SSF analysis tool will be the Robotics Task Analysis Environment (RTAE). This tool will include kinematic and dynamic models of all SSF robots and associated hardware:

- •Mobile Transporter (MT)
- •Mobile Servicing System (MSS)
  - ◊Space Station Remote Manipulator System (SSRMS)
  - ◊Special Purpose Dexterous Manipulator (SPDM)
  - ◊Mobile Remote Servicer Base System (MBS)
  - ◊MSS Maintenance Depot (MMD)
- •Japanese Experimental Module Remote Manipulator System (JEMRMS) comprised of:
  - ◊JEM Main Arm (JEMMA)
  - ◊JEM Small Fine Arm (JEMSFA)
- •SRMS

RTAE will be an integrated analysis tool which can perform kinematic, clearance, and dynamics work. Using RTAE, an analyst will be able to export all the relevant parts of one type of analysis into another, e.g., joint angles, trajectories, initial flight load software, and initial simulation settings. RTAE can perform kinematic and dynamic analyses for manipulators with N degrees of freedom. As a result of these analyses, RTAE can generate software parameters needed by onboard SSF robotic systems. At each step in these analysis process interfaces, RTAE will provide reporting and quality assurance checking of results. RTAE will also evaluate flight and simulation data.

RTAE will utilize powerful graphical simulation technology to enable visual displays of an analysis. If an operation (e.g., extremely complicated dynamics analysis) is not especially interactive and/or requires extensive Central Processing Unit (CPU) time (and thus would be extremely slow to watch), RTAE will allow an operator to conduct an interactive operation in the foreground while batch processing activities proceed in the background.

RTAE will also be used to develop initial procedures. In this capability, RTAE will be used to develop timelines and to analyze crew workloads, intra-vehicular activity (IVA) workstation usage, worksite lighting, visual cues, and video requirements. Once procedures are verified, RTAE can output robot task commands and trajectories for automated sequences in an uplinkable, executable format for use by on-board systems.

3.2 Robotics Task Library

A second tool for the SSF robotics analyst is the Robotics Task Library (RTL). This is a library or database of task data which includes:

1. Task name
2. Task description
3. Robots and tools used
4. Stage of task development
5. Resources required or estimated
6. Constraints
7. Station configuration/increment(s) with which the task is associated or approved
8. Trajectory and procedural data or auto sequence command files used in executing the task.

The tasks included in RTL may have been previously executed, previously planned, or in the process of planning.

RTL is totally compatible with RTAE. No manual translation will be required to transfer RTL data into RTAE. RTAE planners can scan through the RTL task listings by the eight or more categories. If the search by category indicates a possible match or similarity to a task being planned, RTAE users can import the task data file. Once the data is within RTAE, it can be used to drive RTAE's graphical simulators, thus providing the analyst with a visualization of the task. If the task can be re-used, RTAE analysts can "cut and paste" all or part of the RTL data file into RTAE. Any modifications can then be performed within RTAE. Because many SSF robotics maintenance tasks may be very repetitive, such a utility is necessary to streamline the planning process.

3.3 GRAMPS Library

The third tool necessary to planning, visualization, and interactive and graphical analysis is the SSF Graphical And Mass Properties (GRAMPS) Library. This library is a database of all objects at the ORU or structural element level on SSF, stored in a format that can drive graphical simulations. Each of these "base objects" can be accessed by any combination of the following categories:

1. object name or key word
2. location [e.g., all objects within a given volume of space around a given point (x,y,z)]
3. station subassembly name
4. desired increment
5. logistics control number
6. pictorial representation from a catalogue of objects
7. any combination of the above.

GRAMPS object models must be accurate to within one-half inch and one degree for detailed clearance analysis and collision avoidance. The types of data that will be maintained in GRAMPS for each base object include:

1. object identifier
2. physical dimensions
3. surface properties
    a. material
    b. texture
    c. color
    d. albedo
    e. locations of guides and markings
4. mass properties
    a. mass
    b. center of mass
    c. moments of inertia
5. interface definitions
    a. location (relative to the origin)
    b. size and shape
    c. type (e.g., electrical, fluid, structural, grapple)
    d. gender (male, female)
    e. flow direction (e.g., input, output, bi-directional, n/a).

For objects which are comprised of base objects, each layer of a model above the base level contains the following data:

1. list of objects which are components of the layer
2. position and orientation of each object in the layer relative to the origin for that layer (mating of interface points)
3. dynamic envelope of each object in the layer relative to the origin for that layer
4. physical dimensions of the layer if viewed as a single object

5. mass properties of the layer if viewed as a single object
    a. mass
    b. center of mass (relative to the origin for the given layer)
    c. moments of inertia
6. interface definitions
    a. location (relative to the origin)
    b. size and shape
    c. type (e.g., electrical, fluid, structural, grapple fixture)
    d. gender (male, female)
    e. flow direction (e.g., input, output, bi-directional, n/a).

Besides SSF, robotics analysts require models of other objects. These include the Space Shuttle, payloads, any free flyers which will interface with robotics devices, and EVA crewmen.

RTAE will utilize the graphical models within GRAMPS for detailed clearance analysis. Without GRAMPS, robotics analysts will be unable to see the tasks they are planning or to assess clearances.

4.0 Canadian Analysis/Procedures Development Tools

Because SSRMS, SPDM, MBS, and MMD, unlike the SRMS, will be owned by Canada, Canadian Space Agency (CSA) has invested considerable capital and effort into ensuring the existence of analysis, procedures development, and training facilities. Those Canadian Ground Segment facilities include the Procedures Management System (PMS) and the Manipulator Development Simulator Facility (MDSF).

The current Mission Operations Directorate baseline concept is that normal operations procedures development will be performed in the SSCC. CSA/Spar will provide all procedures which are system-specific or which involve envelope expansions of the Canadian robots.

The first step in the development of the Operations Date File (ODF) is the analysis process. Once analysis has been completed, RTAE, through its links with the SSCC procedure development tool, will generate a set of integrated, increment-specific task procedures. These procedures will incorporate system-specific procedures for both Canadian arms developed by CSA/Spar Aerospace.

System-specific procedures include generic and increment-specific operating procedures, system malfunction procedures, and task primitives (lift, rotate joint X degrees, etc.). Generic procedures include power-up, power-down, checkout, etc.

Increment-specific procedures must be closely coordinated with corresponding SRMS Flight Data File procedures for situations such as hand-offs. They must also be coordinated with other SSF mission planners, e.g., Guidance, Navigation, and Control; Operations Planning; Maintenance, Inventory, Logistics Planning (MILP); Extra-Vehicular Activity Systems (EVAS); Communications and Tracking (C&T), etc.

The integrated US-Canadian SSCC robotics planning team will utilize the RTAE to perform analysis and build preliminary integrated procedures Other simulators which will be used to further develop, validate, verify, and train personnel in robotics integrated procedures include:

- Mobile Remote Manipulator Development Facility (MRMDF) - JSC
- SES -JSC
- Shuttle Mission Training Facility (SMTF) -JSC
- Space Station Training Facility (SSTF) -JSC
- MDSF -CSA
- PMS - CSA

Ideally, RTAE will be able to electronically exchange data files with the Canadian PMS. At worst, the two systems should be capable of importing and translating data from each other into a usable format. Once the two planning systems have established a basic set of task procedures through iterative activities, MRMDF, SES, and MDSF will be used to further test, develop and validate them. SMTF, SSTF, and MDSF will be used for final procedure verification and crew/mission controller training on the final procedures. Once the procedures are finalized, they will be generated as an official segment of the Operations Data File using SSCC procedures tools and the RTAE. Figure 1, Robotics Procedures Development Process, indicates this process flow.

5.0 Summary and Conclusion

Today, planning for a complex SRMS mission requires nearly 5,000 manhours. Part of the reason for this heavy burden is the disconnected nature of the tools. SSF cannot sustain continued labor-consuming analysis methods.

Therefore, requirements for a suite of tools designed to streamline the planning process in the SSCC have been written. These tools are the Robotics Task Analysis Environment, Robotics Task Library, and Graphical And Mass Properties Library. These tools will enable robotics task planners to input data from one type of analysis into another, thus reducing re-entry of data and streamlining the planning process. The Robotics Task Analysis Environment will be the primary planning tool; it will be graphical, interactive, and integrated. It will operate with the Graphical And Mass Properties Library to graphically display the results of task analysis and show detailed clearances. The Robotics Task Library will enable re-use of entire tasks or portions of them to build new tasks.

However, robotics planning will not be isolated within the SSCC, as will other types of mission planning. Because SSRMS and SPDM are Canadian systems, analysis and procedures development will involve both the SSCC and relevant facilities of the Canadian Ground Segment, especially the Procedure Management System and the MDSF. Specific details of the procedure development process have yet to be worked out; this paper has indicated the Mission Operation Directorate's baseline concept for international, integrated SSF robotics analysis, planning, and procedure development.

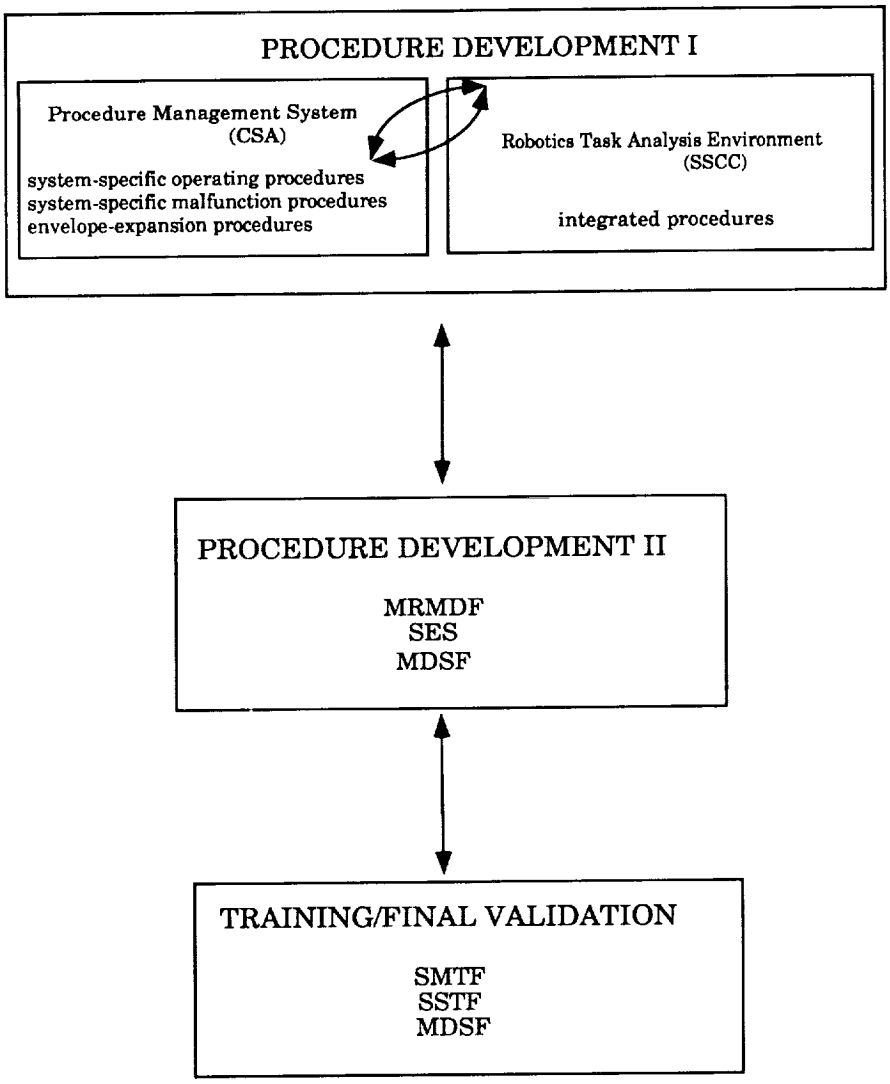


Figure 1 - Robotics Procedures Development Process

References

1. Mann, Laura S./Johnson Space Center/DF44, "Robotics Space Station Training Facilities/Shuttle Mission Training Facility Loading: Response to Action Item SWT 1-F10," July 13, 1990

2. SPACE STATION CONTROL CENTER USER DETAILED FUNCTIONAL REQUIREMENTS (JSC 13193), PRELIMINARY, October 1, 1990

3. SPACE STATION CONTROL CENTER USER DETAILED FUNCTIONAL REQUIREMENTS (JSC 13193), BASELINE, April 8, 1991

4. Thompson, Kelly/MSC, "Robotics Model Assessment Team Minutes," 10/11/90-12/20/90

5. Testa, Bridget Mintz/Barrios/Johnson Space Center/DF44, "Section 5.13 Robotics Planning," SPACE STATION FREEDOM BASELINE OPERATIONS PLAN, JSC 32083, Baseline Version (in progress)

6. Zaguli, Ronald J./Johnson Space Center/DF44, "Robotics Planning and Mission Design Briefing," July 18, 1990

HABITAT AUTOMATION

Rodney E. Swab
NASA Ames Research Center
MS 244-18 Moffett Field, CA 94035-1000

ABSTRACT

A habitat, on either the surface of the Moon or Mars, will be designed and built with the proven technologies of that day. These technologies will be mature and readily available to the habitat designer. We believe an acceleration of the normal pace of automation would allow a habitat to be safer and more easily maintained than would be the case otherwise. This document examines the operation of a habitat and describes elements of that operation which may benefit from an increased use of automation. Research topics within the automation realm are then defined and discussed with respect to the role they can have in the design of the habitat. Problems associated with the integration of advanced technologies into real-world projects at NASA are also addressed.

INTRODUCTION

A habitat, on either the surface of the Moon or Mars, will be designed and built with the proven technologies of that day. These technologies will be mature and readily available to the habitat designer. We believe an acceleration of the normal pace of automation would allow a habitat to be safer and more easily maintained than would be the case otherwise. Because only mature technologies will be useful to habitat designers, it is necessary to assess the current maturity of automation. "Automation", as used in this document, refers primarily to the advanced software control of a complex array of equipment and sensors, and secondarily to the isolated operation of an autonomous robot. A "habitat" is defined to be the shirt-sleeved living and working quarters of a crew in a hostile space-based environment. The specific habitat under consideration is that defined as Option 5A in the *Habitation and Human Systems Addendum* [1] to the *Report of the 90-Day Study on Human Exploration of the Moon and Mars* [2]. This paper examines the operation of a habitat and describes elements of that operation which may benefit from an increased use of automation. These elements include fault-tolerance, graceful degradation, localization of failures, human-machine interaction, non-invasive repair strategies, and some logistics matters. Some research topics within the automation realm are then defined and discussed with respect to the role they can have in the design of the habitat. These topics include fault-diagnosis and recovery methods, planning, and speech-recognition. The research topics are discussed only with reference to their potential application to the habitat design. More detailed sources of information on specific topics are at times suggested.

LOGISTICS

Logistics involves the design, operational planning, and provisioning for the habitat. Logistics has historically taken a back seat to most other disciplines during development projects. In the creation of a remote space-based habitat, however, we can ill afford to have it remain there. Many details need to be decided upon in creating a viable logistics support plan for a space habitat. One of the largest decisions involves the placement of the depot facility. Having a depot facility co-located with the habitat may prove to be a necessity due to the great number of line replaceable units (LRUs) that would otherwise have to be present at the site. Co-locating a depot-level repair facility with the habitat, however, will be a unique undertaking. In the US armed forces, for the support of millions of pieces of electronic equipment, there exist a handful of depot level repair facilities. The habitat depot will, however, fulfill a much more exclusive support role than a general purpose depot.

Regardless of depot placement, the difficulty in maintaining logistics support for the habitat will be great. Logistics support for Operation Desert Storm has been a topic of conversation in recent months. The inability to get the proper supplies to our armed forces in the Persian Gulf could have resulted in unnecessary loss of life. Our inability to get the proper supplies to a space habitat could do likewise. Whatever logistics plan we decide on will have to accommodate complex, risky, and inherently unreliable resupply missions.

It is important to design the subsystems of the habitat to capitalize on commonality. Having different subsystems with interchangeable LRUs, if possible, would add depth and flexibility to system maintenance. An overriding concern of this design process will be to keep unique parts counts at a minimum. This will reduce the cost of initially outfitting the habitat's repair facilities and reduce the ongoing life cycle costs associated with resupply. Perhaps a logistics "Tiger Team" could be assembled whose purpose would be to minimize the number of unique components incorporated into the subsystem designs, as well as to minimize the support items required to maintain the subsystems. Without such an influence it is far too easy for subsystem designers to use either what is available to them or what they are most comfortable with.

DESIGN

Each of the subsystems of the habitat will be integral in sustaining life. This will require constant availability of each of the subsystems. The ultimate design goal of a continuously operating system is for no one failure to incapacitate or degrade system performance. An equally laudable repair goal is for no one repair to require a suspension of, or degradation of, system performance. Fault tolerant system operation, graceful system degradation, and non-invasive repair strategies will be necessary elements of the design of the habitat.

Constant availability implies the presence of fault redundant operation (to ensure stable operation of habitat functions); localization of failures (to prevent cascading effect of failures); subsystem isolation (to prevent cascading effect of failures between subsystems); and an adequate logistics pipeline (to allow for repair of faulty equipment in a timely manner). In a terrestrial factory this presents a difficult, but doable task. On the Moon or Mars it will be much more difficult, with the consequences of failure being much more grave.

Fault Redundancy

Fault redundancy usually takes the form of hot and cold spares and, in some cases, entire backup subsystems. Perhaps some redundancy of this form may be necessary, but an extremely high cost would be paid for it. The cost for launching a metric ton of cargo into orbit is extremely high; and this apart from placement of the cargo on the surface of the Moon or Mars. Because of the conflict between the need for redundancy and its high cost, an analysis is needed to determine the most cost-effective forms of redundancy to employ in the design of the habitat.

Subsystem Commonality

The subsystems, while varying greatly in their duties and designs, provide an opportunity for exploitation of their common features. Each subsystem will be required to 1) plan and control it's own operations; 2) determine it's own ability or inability to function; 3) cooperate with other subsystems as necessary; and 4) communicate with a system level executive as necessary. These points can be restated as 1) planning and scheduling; 2) fault detection, isolation and recovery (FDIR); 3) interfacing; and 4) hierarchical communication and control. Planners and schedulers, unique to each subsystem, should be able to communicate with one another with ease in that they will be required to operate on similar types of objects. Fault diagnostic programs should operate on similar principles so as to benefit one another during their operation. The philosophy behind subsystem interfaces should be defined early and adhered to strictly during the design of the habitat. Subsystem interfaces should be thought of as part of the system and *designed in* as opposed to being patched in when found necessary. Design documents should be created which encourage and enforce design

commonality and consistency in subsystem interfacing. These principles, applied correctly, should result in cooperation of the subsystems at the system level.

Physical Dispersion of Subsystems

While we want the subsystems to use common parts and to be built with similar design paradigms, they will also have to remain electronically isolated from one another to prevent the possibility of failures leaping across subsystems. To minimize the potential hazardous effects physical phenomena, such as fire or flooding, may have on the equipment, the subsystems themselves should also be distributed over a large physical area.

Importance of System "State" Maintenance

Repair strategies must be developed that take into account the state of the equipment and potential loss of state information via failures. "State" refers to the sum of the many facts which together define a point in time in the life of the habitat. It is important that this abstract state represent the true state of the habitat environment as best it can. A sophisticated network of sensors of all types will help us to maintain this state validity. This abstract state will then be used by the various health monitoring systems and fault detection programs to assist in the determination of proper equipment and subsystem operation. It provides a model of the habitat with which the various subsystems can reason. Reasoning upon this abstract state is known as Model-Based Reasoning (MBR).

Each of the subsystems will have different levels of automated response, implemented at the hardware level, to ensure the security of both equipment and personnel. In design terminology, this lowest level of automated response to the detection of hardware faults, is known as *safing the system*. The impact of safing will first be felt, via the sensors, by a system executive. (We use the term Habitat or Space Habitat Executive (HE/SHE). HE will be used for consistency). The system state will then be assessed and recovery procedures, appropriate to the situation, implemented. Perhaps HE could be sent a "heads-up" message just prior to a subsystem safing itself. In this way the strategy used in resuming operation could have been designed beforehand and thus resumption of processing could proceed in a more studied fashion.

This implies a tight coupling of the hardware and software. This tight coupling can only exist if it has been designed into the system. To accommodate this we should ensure a tight communication exists between the hardware and software developers during the design stages. A lack of communication would lead to complication of the software designs and a reduction in the efficacy of the software in handling unusual states.

Human-Machine Interaction

Input/Output (I/O) in the habitat will take many and varied forms. Traditional computer input via the keyboard and output via the computer screen will be assisted by speech recognition systems and natural language understanding capabilities; visual control, possibly with the aid of head gear such as is used in virtual reality testbeds; hearing, via our ability to distinguish variations in tones, as well as the location in three-dimensional space of the source of said tones.

It should prove beneficial that the computer react to the voice of the habitat occupants. Voice input will allow a tangential task to be started while not disrupting the primary task to which the speaker is employed. A driving principle behind the development of the habitat's voice and data control systems, should be the desire to not enslave the habitat occupants to use of a stringent syntax. Stringent enforcement should be needed only when commands are ambiguous or nonsensical.

Visual "heads up" displays, such as are now used in jet cockpits, and flat wall displays and touch screens can be used to free the habitat occupants from the need to sit in front of small computer screens. Virtual reality helmets can be used daily to simplify the control of robots outside of the habitat.

We have only begun to explore the concept of using hearing in human machine interaction. The auditory sense can provide an alternate route for critical information in complex environments during periods in which the

user's visual capacity is already greatly taxed. "Ames is currently investigating the underlying perceptual principles of auditory displays and is also developing a prototype signal processor based on these principles. Rather than use a spherical array of speakers, the prototype maximizes portability by synthetically generating 3-D sound cues in real-time for delivery through earphones. Unlike conventional stereo, sources will be perceived outside the head at discrete distances and directions from the listener. This is made possible by numerically modelling the effects of the outer ears on the sounds perceived at various spatial locations." [3] These discoveries, and the discoveries of other related programs, are rapidly expanding the role of hearing in the design of future man-machine systems.

The occupants of the habitat should be considered the equivalent of jet aircraft pilots for the purposes of habitat design. This is because, like jet aircraft pilots, they will be hard-pressed during critical events to absorb all of the information available to them. This is especially true if we do not attempt to better distribute the sensory load over all of the senses. Because of this similarity we should assess the state of aircraft cockpit design. Advances in the area of jet aircraft human-machine interaction should be given serious consideration in the design of the habitat. Ames Research Center is now preparing a document which addresses the rationale and philosophy of human-centered aircraft automation. It will address the issues posed by aircraft automation as it has evolved over the past sixty years [4].

Having multiple forms of I/O also provides an inherent redundancy and flexibility in day to day operational use and control of the habitat. When one form of control is incapacitated, for whatever reason, the chances would be much greater that another I/O route exists to fulfill a requirement.

OPERATION

The subsystems of the habitat will interact constantly. In large systems, subsystem interaction is often handled on an individual basis, with interfaces between subsystems being defined as needed. Within the habitat we will need to constrain subsystem interactions to isolate them from one another. This isolation will allow for 1) the independent development of FDIR programs for each subsystem; 2) the development of a system-wide health monitoring and fault diagnostic program; and 3) the isolation of failure effects to the subsystem in which the failure occurs.

An example of the need to coordinate the activity of subsystems is exemplified by the direct correlation between the consumption of power and the generation of heat. The load placed on a thermal cooling system is driven by the generation of heat which is a side-effect of power usage. As power usage increases the need for thermal cooling increases. When, however, thermal cooling capacity is diminished in some way it may be necessary to reduce the generation of heat, which can only be done by eliminating non-essential power usage. It is the automation of such subsystem interactions that would help simplify the lives of the habitat occupants.

Sensors will maintain a constant flow of real-time information to the individual subsystems of the habitat as well as to HE. HE will resolve the conflicts between subsystems' individual plans and schedules in accordance with a greater awareness of the proper subsystem roles in the habitat. This arbitration between subsystems is not intended to layer a bureaucracy on the running of the habitat. Having HE arbiter all subsystem interaction would produce an unacceptable bottleneck in operation and would increase the damage potential of single point failures. It's role here would be strictly that of resolving subsystem conflicts.

Another aspect of the role of HE involves the scheduled interaction of humans with equipment. Almost all human activity impacts equipment resources in some way. HE will be responsible for the control and sharing of these system resources.

MAINTENANCE

With manpower being the most precious resource of the habitat, both morally and financially (some figures place the hourly cost of having astronauts in orbit at $35,000 [5]), we must design the subsystems of the habitat to function as autonomously as is practicable.

The normal operation, detection and diagnosis of faults, and repair of subsystems, should be automated to the greatest extent possible. Subsystem health monitoring should allow for automated recognition of, and rerouting of subsystem operation around, minor faults without impacting system operation. Health monitoring should also provide automated diagnosis of minor faults which do impact subsystem operation as quickly as possible. Defective LRUs will be replaced by human repairmen and either disposed of or forwarded to the depot for repair. Over time, perhaps automated subsystem assistants can help with LRU replacement and disposition.

Robotic Depot Repair Facility

An automated depot-level repair facility is being considered for the habitat. This could be realized by developing a highly automated facility for the repair of all repairable LRUs. Automated component-level equipment repair can be facilitated by such things as 1) human staging and previewing of LRUs; 2) Optical Character Recognition (OCR) coding of all LRUs and replacement part storage locations; 3) recording of all LRU component placements; and 4) a highly constrained environment within which the robot can perform the repairs. This repair strategy will require extensive development and may need to be phased in at an operational habitat.

Active and Passive Maintenance

System maintenance will have a passive and an active element. The passive element can be thought of as a health monitoring system. The role of this system is to minimize the number of malfunctions requiring immediate operator attention. The system should be capable of handling the great majority of malfunctions, thus allowing the habitat occupants to perform other, more critical tasks. It incorporates 1) trend analysis which can lead to preventive maintenance tasks being assigned to prevent future failures; 2) automatic reconfiguration of system elements to bypass suspected failed LRUs; 3) control of fusion of sensors and static data displays to keep the habitat occupants informed of system status; and 4) interactive data displays to inform the more inquisitive user of the state of the habitat.

Active maintenance will be in the form of FDIR programs, unique to each subsystem, capable of fault-isolation to the LRU level. The FDIR programs will be developed independent of one another but with a common design methodology to allow for communication in the larger system-wide FDIR program. The subsystem FDIR programs will communicate with one another via a blackboard data architecture, controlled by HE, allowing them to share information vital to one another. Using a knowledge-based systems approach, the same inference engine design for each of the subsystems can be used while allowing the data unique to each subsystem to determine the troubleshooting path. This will facilitate a more tractable validation and verification of the various FDIR programs and help simplify the development of a system-wide FDIR program.

The transition from passive to active maintenance will at times take the form of responses to status updates (in the case of non-critical failures) or responses to alarms (in the case of critical failures). HE will be fed information from each of the subsystems at specified time intervals, as well as asynchronously in the event of anomaly detection or user interaction.

APPLICATION OF AUTOMATION

Candidates for automation, of either form defined earlier, are those tasks which are 1) time consuming; 2) repetitive; 3) uninteresting; 4) well-defined and highly constrainable; and/or 5) operate in isolation. This is not a definitive list in determining what to automate, but does serve as general guidance when considering candidates for automation.

Automation candidates already mentioned include the health monitoring system, HE, the FDIR programs, planners and schedulers unique to each subsystem, and the robotic depot repair facility. Other tasks, which might lend themselves at least partially to automation, are "household chores", such as, food preparation and cleanup, vacuuming and dusting, storage and access of work area tools, inventory control and replenishment, waste disposal, and bathroom cleaning.

As the habitat is to be manned continuously, a means of locally producing fresh vegetable produce will be necessary. A "salad machine", that will provide a variety of fresh vegetables for astronauts on long voyages, is now operational at NASA Ames. [6] The near-term goal is to provide astronauts of Space Station Freedom with fresh salads. The machine may also provide a beneficial side-effect in improving the morale of the crew by offering them a creative outlet during their free time, such as is provided by tending a garden on Earth.

Many opportunities exist for the inclusion of automation in a space habitat. A "a day in the life of the habitat" scenario, developed early on in the project, would provide a model upon which automation concepts could be modeled and thus compared against their more traditional counterparts. The model would also provide an environment within which the interaction of tasks effected by automation could be assessed.

RESEARCH

The following research topics have been referenced previously in the text. Appearing here is a brief description of their current capabilities and their weaknesses with respect to our application of them in design of a habitat.

Speech Recognition

Speech recognition is the capacity for a computer to "hear" and correctly identify the spoken word. "Few applications of speech-recognition technology have reached beyond simple, speaker-dependent, isolated-word recognizers with vocabularies of a dozen to a hundred words. Small vocabularies and poor accuracy have limited the applications suitable for speaker-independent systems." [7] Though this was stated over four years ago, and much progress has been made since that time, the more successful systems still recognize only isolated words or short phrases, and require "training" to recognize each speaker's voice. There is also usually no "understanding" of the words spoken (although it can be argued that this is an extension to the concept of speech recognition). The words are usually used only as dumb commands, without semantic significance, in the execution of predefined actions. Further research needs to be performed to improve upon speech recognition in the areas of continuous-speech, speaker-independence, vocabulary size, and accuracy. SRI International has been working on a continuous-speech, speaker-independent, 20,000 word speech recognition system for DARPA. [8] This system, when completed, should be evaluated with respect to its potential usefulness in the habitat.

Natural Language

Natural language can also be thought of as *speech understanding*. Speech recognition identifies the words, but natural language understanding attempts to derive meaning from the words. This meaning is needed, on the computer side, to "understand" what it is being commanded to do. In addition, the computer needs to be capable of generating semantically correct replies for the user. Natural language is now advertised to be resident in many newly released software products. The natural language referred to is usually nothing much more than lazy syntax enforcement in combination with COBOL-like command statements. While this is in itself a useful software concept, it is not what we have defined here as natural language understanding. To be useful in the operation of the habitat, much more research in natural language understanding needs to be done. At this stage we may be better served by integration of the current, more popular, meaning of natural language.

Model-based Reasoning (MBR)

"Model-based reasoning uses an internal symbolic model of the system of interest and updates the state of this model based on sensor evidence and cause/effect analysis." [9] MBR has several advantages over other forms of fault diagnostic systems. It can handle systems too large for traditional troubleshooting procedures developed in conjunction with a Failure Modes and Effects Analysis (FMEA). It can also lead to the discovery of faults other than those for which it has been proven to work. "The model-based capabilities of TEXSYS were shown to be advantageous, particularly for detection of unforeseen faults and sensor failures." [10] It may also require less of a knowledge engineering effort, as the model is based more upon device behavior and

less upon heuristics unique to the domain. MBR has its drawbacks, however, one of which is excessive use of processor time, as shown in the following excerpt. "It was necessary to rely on a classification or rule-based approach to interpret conflicts in the expert system's model, given the slowness of structural *(model-based)* reasoning." [10] Another weakness of the MBR approach is the time lag which exists in updating the model to reflect what has happened in reality. Often, during times of increased activity, valid states are interpreted as error states due to the model having been inconsistent, for perhaps only a moment, at the time that the data were sampled. Research in MBR needs to focus on how to better represent the actual state expected in the model and how to reason in more general, domain independent ways.

Planning

"Any intelligent system that operates in a moderately complex or unpredictable environment must be reactive, that is, it must respond dynamically to changes in its environment." [9] Planning is one activity which we hope to have migrate to the computer in great part due to its time-consuming nature. If done manually, in the habitat, little time would be left for other activities. The current state of planners, however, does not support the depth and flexibility required of the planners in the habitat. Planners are incapable of working on general problems and may continue to be for some time. Successful planners sometimes work only within oversimplified domains or are very domain specific. Planning in the midst of a dynamic domain also changes the content of the plan continually, often invalidating it before it is complete. Much more basic research in planning in a dynamic domain needs to be performed. Perhaps, for habitat needs, research should focus on human assisted planning in addition to the more popular autonomous planning. Some excellent suggestions on space-based planning have been made in Reference 5 (pages 4-8 thru 4-10).

CONCLUSIONS

The following quotations are all taken from a MITRE Report entitled, *Space Station Freedom Program Advanced Automation: Volume I* [11] and are referenced here without additional comment.

"The research community can be characterized as Ptolemaic: advanced automation is the center of their universe, the rest of the universe orbits around this center." *page 6*

"A Copernican view of advanced automation is required if these technologies are to be used within operational applications." *page 7*

"...the technology used for Apollo and Shuttle was successful and should be good enough..." page 8

"Too much innovation causes disruption, while excess stability creates stagnation. There is currently no environments for transitioning innovative technologies and applications into the stable production environments." *page 11*

"... there is a distinct gap that must be filled between the relatively unconstrained environments of the test beds and the constrained production facilities and operations environments." *page 14*

Suggestions for Future Consideration

Throughout this paper questions have been raised and further studies have been suggested. To again highlight them they appear here in bullet form.

- To assist in incorporating automation into the operation of the habitat, we must make the habitat designers aware of the areas which may benefit from automation.

- An analysis is necessary to determine the most cost-effective form of redundancy to be employed in the design of the habitat.

- Perhaps a logistics "Tiger Team" could be assembled for the purpose of maximizing commonality and minimizing the number of unique components incorporated into the subsystem designs.

- The location of the depot repair facility will provide the foundation for making many future habitat design decisions.

- Subsystem interfaces should be *designed in* and not developed ad hoc.

- Planners, schedulers, and FDIR programs,

unique to each subsystem, should use common data structures to ease system communication.

• Habitat hardware design should consider the effect *safing* will have on HE.

• Having I/O take varied forms will provide an inherent redundancy in the day to day operational use and control of the habitat.

• To assist in the assessment of automation scenarios perhaps a "day in the life of the habitat" model could be developed.

• Perhaps NASA could address the lack of suitable environments for the transitioning of innovative technologies and applications into stable production environments.

REFERENCES

1. NASA Committee, "Habitation and Human Systems Addendum to the 90-Day Study on Human Exploration of the Moon and Mars," NASA Report, Washington, DC, November ,1989.

2. NASA Committee, "Report of the 90-Day Study on Human Exploration of the Moon and Mars," NASA Report, Washington, DC, November, 1989.

3. Wenzel, Elizabeth M, "3-D Auditory Display Systems Research," Aerospace Human Factors Research Division Information Brochure, NASA Ames Research Center, Moffett Field, CA, August, 1990, Page 23.

4. NASA Ames Research Center - Code FL, "Aerospace Human Factors Research Division", NASA Brochure, NASA Ames Research Center, CA, August, 1990, Page 50.

5. NASA Committee, "Space Station Freedom Automation and Robotics: An Assessment of the Potential for Increased Productivity", NASA Report, Marshall Space Flight Center, AL, March, 1990, Page 4-3.

6. Hutchison, Jane, "Fresh Veggies for Space Flight," NASA Activities, NASA Ames Research Center, Vol., 21, No., 5, September/October, 1990, Page 17.

7. Wallich, Paul, "Putting speech recognizers to work," IEEE Spectrum, April, 1987.

8. Murveit, Hy, "Real-Time Speech-Recognition Systems," DARPA Technical Proposal, Contract No: N00039-85-C-0302, October, 1988.

9. Rockwell Committee, "Research On Advanced Engineering Software For In-Space Assembly and the Manned Mars Spacecraft," Rockwell International Corporation, June, 1989.

10. Glass, Brian J, "Results of the Systems Autonomy Demonstration Project," NASA Ames Research Center, Moffett Field, CA, October, 1990.

11. Bayer, Steven E, "Space Station Freedom Program Advanced Automation: Volume I," The MITRE Corporation, December, 1989.

ROLE OF AUTOMATION IN THE ACRV OPERATIONS

S. F. SEPAHBAN

ACRV Mission Support and Ground Operations Manager,
Assured Crew Return Vehicle Project Office
NASA/Johnson Space Center

The Assured Crew Return Vehicle (ACRV) will provide the Space Station Freedom with contingency means of return to earth 1) of one disabled crew member during medical emergencies, 2) of all crew members in case of accidents or failures of SSF systems, and 3) in case of interruption of the Space Shuttle flights. A wide range of vehicle configurations and system approaches are currently under study. The Program requirements focus on minimizing life cycle costs by ensuring simple operations, built-in reliability and maintainability.

The ACRV philosophy of embedded operations is based on maximum use of existing facilities, resources and processes, while minimizing the interfaces and impacts to the Space Shuttle and Freedom programs. A preliminary integrated operations concept based on this philosophy and covering the ground, flight, mission support, and landing and recovery operations has been produced.

To implement the ACRV operations concept, the underlying approach has been to rely on vehicle autonomy and automation, to the extent possible. Candidate functions and processes which may benefit from current or near-term automation and robotics technologies have been identified. These include, but are not limited to, built-in automated ground tests and checkouts; use of the Freedom and the Orbiter remote manipulator systems, for ACRV berthing; automated passive monitoring and performance trend analysis, and periodic active checkouts during dormant periods. The major ACRV operations concept issues as they relate to the use of automation will be discussed.

N93-11978

Decision Rules for Spaceborne Operations Planning

Submitted for Presentation at
Space Operations, Applications, and Research Symposium (SOAR 91)

NASA/Johnson Space Center
Houston, Texas

July 9-11, 1991

Jeffrey H. Smith†

ABSTRACT

Recent study of Space Station *Freedom* requirements for extravehicular activity (EVA) to perform external maintenance tasks emphasize an oversubscription of resources for performing on-orbit tasks. Extravehicular robotics (EVR) and cooperative EVA combined with EVR (using crew and robots synergistically to perform tasks) have been suggested as a part of the solution to reduce EVA. The question remains however, "Under what conditions is it cost-effective to use the EVA and/or EVR resource?" The answer to such a question also has implications for the Space Station *Freedom* and its external maintenance as well as the Space Exploration Initiative (SEI) where the issue of work-system allocation is magnified by the long distances and scope of EVA work.

This paper describes a simple technique of interest to operational planners and robot technology planners for determining in an economic context whether to use EVA alone, EVR alone, or Cooperative EVA. It is also shown that given (1) the task times for these alternatives, and (2) the marginal costs of EVA, EVR, and IVA, the appropriate work system for performing the task can be identified. The paper illustrates how the work-system choice is based on the *ratio* of costs. An example using Space Station *Freedom* data is presented to illustrate the trade-offs among alternative work-systems.

Session A7: ASSEMBLY AND SERVICING

Session Chair: Dr. Charles Wooley

N93-11979

RESEARCH AND DEVELOPMENT AT ORNL/CESAR TOWARDS COOPERATING ROBOTIC SYSTEMS FOR HAZARDOUS ENVIRONMENTS

R. C. Mann, K. Fujimura, M. A. Unseren
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6364
mannrc@ornl.gov

ABSTRACT

One of the frontiers in intelligent machine research is the understanding of how constructive cooperation among multiple autonomous agents can be effected. The effort at the Center for Engineering Systems Advanced Research (CESAR) at the Oak Ridge National Laboratory (ORNL) focuses on two problem areas: (1) cooperation by multiple mobile robots in dynamic, incompletely known environments; and (2) cooperating robotic manipulators. Particular emphasis is placed on experimental evaluation of research and developments using the CESAR robot system testbeds, including three mobile robots, and a seven-axis, kinematically redundant mobile manipulator. This paper summarizes initial results of research addressing the decoupling of position and force control for two manipulators holding a common object, and the path planning for multiple robots in a common workspace.

INTRODUCTION

Interest in issues related to achieving effective coordination of multiple robotic devices has been growing over the past few years. There are two main areas of active research related to cooperative robots: (1) path planning and navigation for multiple mobile robots sharing a common workspace, and (2) coordinated or cooperative use of two or more robot manipulators to perform a set of tasks. A useful compilation of representative publications on the topic can be found in [1]. Proceedings of the IEEE International Conferences on Robotics and Automation during the past four years also represent a convenient source of information about recent research efforts. References [2] and [3] contain useful bibliographies covering coordinated robot manipulators and path planning for multiple mobile robots, respectively.

There is a broad spectrum of fundamental problems associated with cooperating multiple robots, ranging from high-level planning to low-level control, and to important architectural and systems issues. Solutions to these problems will have great impact on the safety and productivity of operations in a number of applications including flexible manufacturing, nuclear energy facilities, environmental restoration and waste management, future intelligent transportation systems, as well as space and underwater applications.

CESAR was established during FY-1984 at ORNL for the purpose of addressing fundamental problems and issues arising in the development of intelligent machines. The approach at CESAR involves concentrating part of the resources on the development of an evolving series of mobile robot prototypes HERMIES (Hostile Environment Robotic Machine Intelligence Experiment Series). These machines serve as testbeds for new methods, and hardware and software developments. They are used in experimental scenarios that provide the necessary quantitative data for testing and validation of new approaches, as well as for performance evaluation of different robot system components in integrated systems. HERMIES-IIB and -III (see Figure 1) are currently operational mobile robots [4, 5]. HERMIES-III is equipped with the 7 degree-of-freedom CESARm research manipulator. A second redundant manipulator will be available in the near future. A third mobile platform with simultaneous translational and rotational motion capability as well as on-board VLSI fuzzy logic processors [6] is also part of the unique experimental facilities. Among the most recent experiment scenarios for proof-of-principle demonstrations with the HERMIES robots were the autonomous clean-up of simulated chemical spills [7], and autonomous mapping of areas and objects with beta-radiation contamination (manuscript in preparation).

This paper summarizes recent work at CESAR in the areas of decentralized path planning for multiple mobile robots, and control of coordinated manipulators. The current focus of these activities is on multiple agents with heterogeneous capabilities, with respect to sensing, manipulation, mobility, reasoning. Application focus is derived from a number of applied programs that can benefit from this research in several problem areas, including site characterization, construction, remediation of contaminated sites, and decontamination and decommissioning of facilities.

COORDINATION OF MULTIPLE MOBILE ROBOTS

We have been addressing problems associated with global and local motion planning for multiple mobile robots in a common workspace. Previous and on-going work in this area can be broadly characterized into approaches based on centralized or on distributed planning and control. Early references concerning these approaches are [8] and [9].

Most centralized approaches assume complete knowledge of the workspace and capabilities of all the robots, and proceed to develop a plan which is then followed by all robots. It was shown that the amount of computation involved is exponential in the number of robots [10].

Figure 1: The CESAR mobile robots HERMIES-IIB (right) and HERMIES-III (left).

Central planning for all robots in a prioritized order has been employed [11] to avoid this complexity, at the expense of sub-optimal solutions.

In distributed approaches, each robot develops a plan based on information available about motions of the other robots in the workspace. This information can be rather limited, and usually depends on the robot's sensory capabilities and/or on the communication bandwidth among the different robots. Solutions can generally not be guaranteed to be globally optimal, and deadlock situations may occur. Recent reports on distributed approaches include [9, 12, 13] . Among the simplifying assumptions made in many of these contributions is the homogeneity of the robots with respect to mobility, sensing and reasoning capabilities.

Motivated by many potential application areas for multiple mobile robots, e.g., nuclear environments, space, environmental restoration and waste management, we have been working on the problem of decentralized motion planning for multiple heterogeneous mobile robots in a common workspace.

A computer simulation system has been developed in which the model for each robot or agent consists of three modules: a planning algorithm, knowledge about the environment, and an action interval. These modules determine how well the agent can navigate to a destination point based on knowledge about the environment, how much the agent knows about the current status of its workspace, and how quickly the agent can react to changes in the workspace. Each robot maintains a local map of its environment. The scope of a robot's sensor(s) is reflected in this map. The planning module generates a path based on the information in the map. The agents are treated as completely independent without direct communication among them. Motion planning is based either on visibility [9] or on accessibility [14]. A robot computes estimated future locations of other robots and moves in one of the directions along which a future collision can be avoided. Collisions are avoided by changing the direction of motion, while moving at a constant speed. Each robot repeats the process of planning, acting (moving), and updating its map at a frequency determined by the action interval.

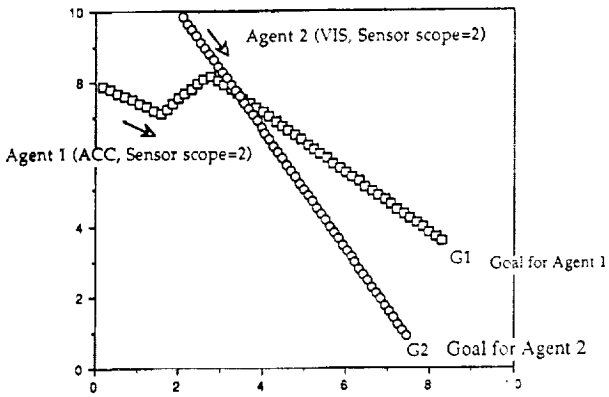


Figure 2. Example of paths followed by two robots with different planning algorithms.

The simulation system was recently described in detail [3]. Among others, our results show that distributed mobile robots can coordinate their motions without deadlocking when appropriately different obstacle avoidance strategies are used by different mobile robots. The simulation system is a useful tool for the investigation of many issues concerning sensing and planning under uncertainty, including reactive and high-level planning for multiple mobile robots.

COOPERATING ROBOT MANIPULATORS

Advancements of our basic understanding of how to accomplish efficient multi-arm manipulation will represent benefits for many application areas for advanced robots in unstructured environments. The effort at CESAR has focused so far on issues arising when two manipulators hold a common object.

Two cooperating manipulators can lift and transport an object whose mass and/or geometry, (e.g., a long, rigid beam) is beyond the carrying capacity of a single manipulator. Two manipulators can directly assemble two parts into a rigid end-product where each part is held by a manipulator. This typically allows for the reduction or even elimination of the expensive, custom-made fixturing often required for a single manipulator to accomplish the same task. Unfortunately, when two manipulators mutually hold an object, the three form a single closed-chain mechanism and a loss of degrees of freedom(DOF) occurs. Strong kinematic and dynamic interactions between the manipulators due to the shared payload result in a constrained system motion. Clearly the manipulators cannot function independently in this closed-chain configuration.

In order to better understand the problems associated with two manipulators lifting and transporting a common object, an adequate process model describing the dynamic behavior of the constrained system is required. In [2], a rigid body dynamical model has been developed for two structurally dissimilar manipulators holding a rigid object in a three-dimensional workspace. The configuration of the system is shown in Figure 3. The final model consists of two sets of equations. One set constitutes the reduced order model which governs the motion of the closed-chain system. The generalized contact forces imparted to the common load by the manipulators are eliminated from the reduced order model but are calculated separately by the other set of model equations. A nonlinear control architecture consisting of the sum of the outputs of two controllers is suggested, which according to the model, leads to the exact decoupling of the position- and force-controlled DOF during motion of the system. The composite controller enables the designer to develop independent, non-interacting control laws for the simultaneous position- and force-control of the closed-chain system.

The approach given in [2] has been generalized to the case of two manipulators holding two types of complex payloads: (i) a spherically jointed object and (ii) a part containing a revolute joint. In [15], the problems of dynamically distributing the jointed loads and of quantifying and controlling the internal stress, strain, and torsion component of the generalized contact forces which were not addressed in [2] are discussed. The problem of solving the rigid body system model for the forward dynamics(i.e., to determine the output response to given applied inputs) is demonstrated to be well-specified,

whereas the solution of the model for the inverse dynamics (i.e., to determine the required inputs when the desired output response of the system is given) is underspecified. It is shown that the number of configuration DOF lost due to the imposition of the kinematic constraints is the same as the number of DOF gained for controlling the internal stress contact forces which do not induce motion in the shared, jointed payload. The composite control architecture proposed in [15] completely decouples the position- and internal stress force-controlled DOF.

CONCLUSIONS

This report provides a brief synopsis of research at ORNL/CESAR in the area of cooperating multiple robots. Initial efforts have been focusing on decentralized path planning for multiple mobile robots, and on controlling two robot manipulators holding a common object. A simulation system was developed that allows for investigations of different planning strategies for multiple mobile robots that differ with respect to their sensing, mobility, and reasoning capabilities. The system supports the study of different reactive behaviors as well as high-level control approaches. A rigid body dynamical model was developed for two structurally dissimilar manipulators holding a rigid object in a three-dimensional workspace. A nonlinear control architecture was suggested, which according to the model, leads to the exact decoupling of the position- and force-controlled DOF during motion of the system. This approach has been generalized to the case of two manipulators holding two types of complex payloads: (i) a spherically jointed object and (ii) a part containing a revolute joint.

Results and lessons learned form these initial studies are now being transferred into efforts that are more experimentally oriented and are making use of the CESAR mobile robots and manipulator(s) for proof-of-principle demonstrations that are relevant to a number of applied programs.

REFERENCES

1. "Multirobot Systems", Ed. by R. Mehrotra, M. R. Varanasi, IEEE Computer Society Press, 1990.

2. Unseren, M. A., "Rigid body dynamics and decoupled control architecture for two strongly interacting manipulators," Robotica (accepted for publication).

3. Fujimura, K., "A model of reactive planning for multiple mobile agents, IEEE International Conference on Robotics and Automation, pp. 1503-1509, Sacramento, CA, April 1991.

4. Burks, B. L. , deSaussure, G. L., Weisbin, C. R., Jones, J. P. , and Hamel, W. R., "Autonomous navigation, exploration and recognition," IEEE Expert, Winter 1987, pp. 18-27.

5. Weisbin, C. R., Burks, B. R., Einstein, J. R., Feezell, R. R., Manges, W. W., and Thompson, D. H., "HERMIES-IIII: a step towards autonomous mobility, manipulation and perception," Robotics, 8, 1990, pp. 7-12.

6. Killough, S. M. and Pin, F. G. , "A fully omnidirectional wheeled assembly for robotic vehicles," ANSI Vol. 61, pp. 425-426, American Nuclear Society 1990 Annual Meeting Proceedings, 1990.

7. Reister, D. B., Jones, J. P., Butler, P. L., Beckerman, M., and Sweeney, F. J., "Demo89 - the initial experiment with the HERMIES-II robot," IEEE International Conference on Robotics and Automation, pp. 2562-2567, Sacramento, CA, April 1991.

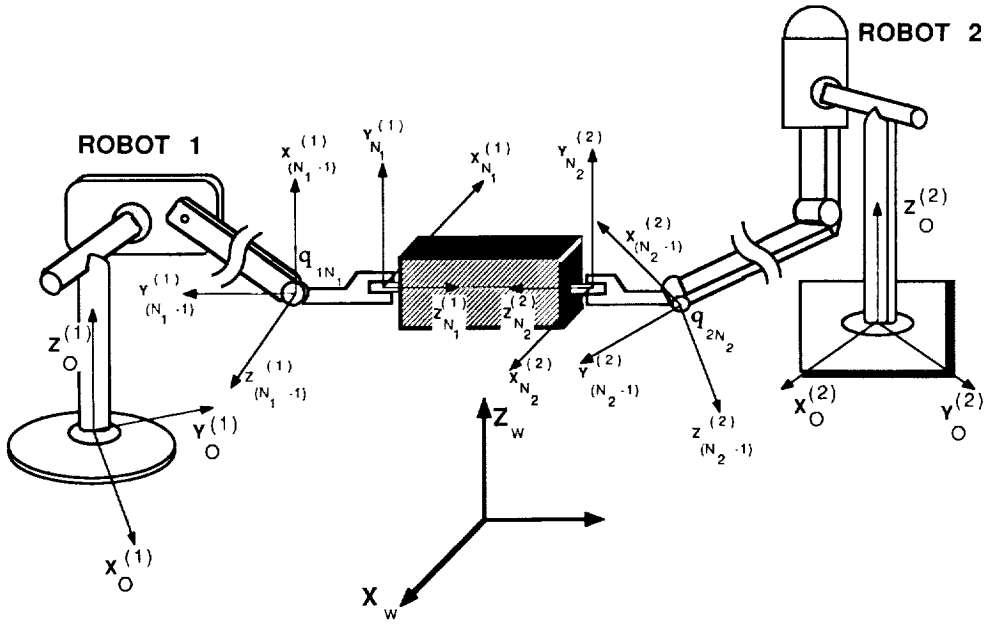


Figure 3. System configuration and coordinate system assignment.

8. Erdmann , M. and Lozano-Perez, T., "On multiple moving objects," Algorithmica 2, 4, pp.477-522, 1987.

9. Tournassoud, P., "A strategy for obsacle avoidance and its application to multi-robot systems," Proceedings of IEEE International Conference on Robotics and Automation, pp. 1224-1229, San Francisco, CA, April 1986.

10. Spirakis, P. and Yap, C., "Strong NP-hardness of moving many discs," Information Processing Letters, 19, pp. 55-59, 1984.

11. Buckley, S. J., "Fast motion planning for multiple moving robots," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 322-326, Scottsdale, AZ, May 1989.

12. Saito, M. and Tsumura, T., "Collision avoidance between mobile robots," Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems, pp. 473-478, Tsukuba, Japan, September 1989.

13. Lee , J.and Bien, Z., "Collision-free trajectory control for multiple robots based on neural optimization network," Robotica 8, 3, pp. 185-194, July-September 1990.

14. Fujimura , K. and Samet, H., "Time-minimal paths among moving obstacles," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1110-1115, Scottsdale, AZ, May 1989.

15. Unseren, M. A., "A rigid body model and decoupled control architecture for two manipulators holding a complex object," Robotics and Autonomous Systems (accepted for publication).

AUTOMATED RESUPPLY OF CONSUMABLES: ENHANCEMENT OF SPACE COMMERCIALIZATION OPPORTUNITIES

Davoud Manouchehri
A.J. Mauceri

Rockwell International Corporation
12214 Lakewood Boulevard, M/S DA25
Downey, CA 90241
(213) 922-2478

ABSTRACT

This paper addresses work performed at Rockwell International's Space Systems Division to investigate the feasibility of, and develop concepts for, automated and/or robotic resupply of consumables on orbit. The work focuses on the resupply of satellites and is described in five sections. First, the various problems relating to resupply on orbit are discussed: for example, economic concerns, fuel handling problems, and safety issues. Next major methods of effecting fuel transfer on orbit are summarized, together with their advantages and disadvantages. Direct fuel exchange is emphasized as the most feasible technique. Third, guidelines are developed for automated/robotic refueling mechanisms to accomplish on-obit consumable resupply. For example, the guidelines cover safety, reliability, maintainability, alignment, induced loads, thermal protection, leaks, extravehicular activity (EVA) interface, and so on. The fourth part of the paper covers the development of design concepts for satellite resupply robotic interfaces that comply with the guidelines. Concepts include servicer fluid transfer system and satellite propulsion system, and a combined docking/umbilical device. Last, future technical development in these areas are discussed.

INTRODUCTION

There are several reasons why on-orbit refueling is a necessary capability. Fuel may be consumed faster than expected: fuel may leak unexpectedly. Unless remedied, either situation could bring about a severe financial loss to the owners of the satellite or to its insurers. In addition, satellites may be built to require refueling after a known number of years, for example, the Gamma Ray Observatory (GRO) is expected to need refueling about 3 years after launch. Most of the proposed Space Exploration Initiative (SEI) assets require on-orbit fueling. Furthermore, if on-orbit fueling capability was available, many more satellites would be designed accordingly to extend their life and reduce usage cost. However, since development of this capability requires substantial funding, it is not feasible to absorb all its cost in one satellite program/project, unless it is specifically designed to address this issue. So far NASA, DoD, and commercial satellite builders have avoided incorporating on-orbit refueling into their requirements on a broad scale. However, we believe it will not be long before the designers of future satellites will incorporate on-orbit refueling, wherever it is economically feasible, to take advantage of its economic and safety benefits.

There are two main methods for accomplishing on-orbit refueling: EVA crew or a remotely operated robotic/automated device that does not need nearby human presence and control. We believe that automated/robotic refueling presents many advantages over EVA methods. The EVA crew may have problems reaching the satellite, even if it is in an EVA compatible orbit. EVA time is an expensive commodity and an EVA refueling excursion would greatly add to a mission's cost. EVA is time limited to about 6 hours. If refueling takes longer, a second EVA team may have to make an EVA excursion. In addition, considerable EVA time is spent in task setup and in translation to the work site. Last, and most important, EVA operations carry a greater risk than intravehicular activity (IVA) operations. The environment is more hazardous, and the propellant, especially hypergolics, impose increased safety risks.

If an automated/robotic system is used to refuel the satellite, a new set of problems must be addressed. Although the human is no longer in close proximity to the refueling operation, safety issues still concern avoiding damage to the satellite by the fueling device and vice versa. Other issues concern the interface between the satellite and the robotic fueling system. For example, markings and color coding must be machine readable, fluid connectors must be compatible with robotic/automated operations and stability/handholds must be compatible with the device's stability system. The compatibility requirements should be embedded as a set of standards or guidelines so that compatibility exists across many classes of satellites that are then serviceable by the same type of automated/robotic device. This paper therefore addresses the problem of providing concepts and guidelines for robotic/satellite interface so that the satellites can be easily refueled by an automated/robotic system.

METHODS OF PROPELLANT TRANSFER

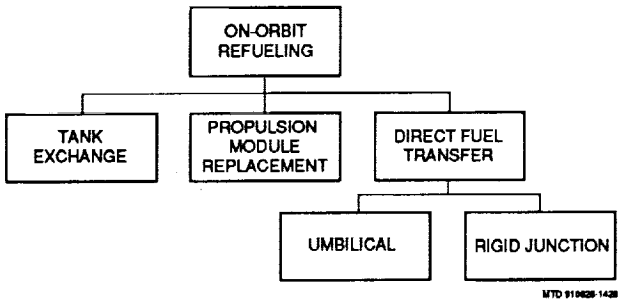
There are three major methods of on-orbit propellant transfer: direct fluid transfer; tank exchange; and propulsion module exchange (Refer to Figure 1).

Direct Fluid Transfer

Direct fluid transfer, as the name implies uses the same concept as a car at a service station: fluid is directly transferred from the servicer tank to the satellite tank.

Tank Exchange

This method involves the replacement of the empty fuel tank by a full fuel tank. The fuel tank is therefore an orbital



"Figure 1 - Metholds of On-Orbit Fuel Transfer."

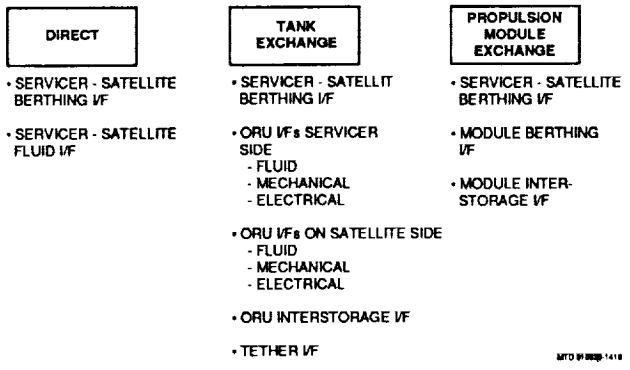
replacement unit (ORU) and must accommodate all the features that allow it to be safely and efficiently handled by a robotic/automated system. For example, it should have an appropriate grappling point. It should have self-guiding and -locking mechanical, fluid, and electrical connectors.

Propulsion Module Exchange

This method requires replacing the entire propulsion system, which becomes an ORU. This method requires the propulsion system to accommodate all necessary features for remote handling. The method permits the removal of a propulsion system for service at a space maintenance facility. The propulsion module will include fuel tank, thrusters, fluid lines and fuel management system.

Trade Study

A series of trade studies were performed to review the merits of each concept. Direct fueling had the least impact on satellite design. It also allowed full utilization of available fuel. Finally, it required the smallest number of interfaces. The disadvantages include longer operation time and safety concerns related to actual displacement of the fuel from one tank to another. Figure 2 shows the required interfaces for each method.



"Figure 2 - Required Interfaces for Fluid Transfer Metholds."

The tank exchange method reduces operation time and lessens the number of interfaces. On the other hand, there are four disadvantages. First, the tank exchange should always take place before the tank is completely empty. This wastes valuable fuel and reduces the overall cost efficiency. Second, a tank needs to be carried to the rendezvous site for the fueling of each satellite, thus the servicer weight will increase significantly. Third, the capability for exchanging tanks imposes too many design constraints on the satellites. Fourth, transfer of such a large mass from the servicer to the satellite and the resulting sudden shift in the center of the gravity raises some concerns regarding control and system stability.

Propulsion module exchange offers the same advantages and disadvantages as the tank exchange option. However, its disadvantages are more pronounced, which resulted in making this option the least favorite.

Based on the results of the trade study, it was determined that direct fuel exchange is the most cost-effective and feasible technique. Therefore, this paper concentrates on this method.

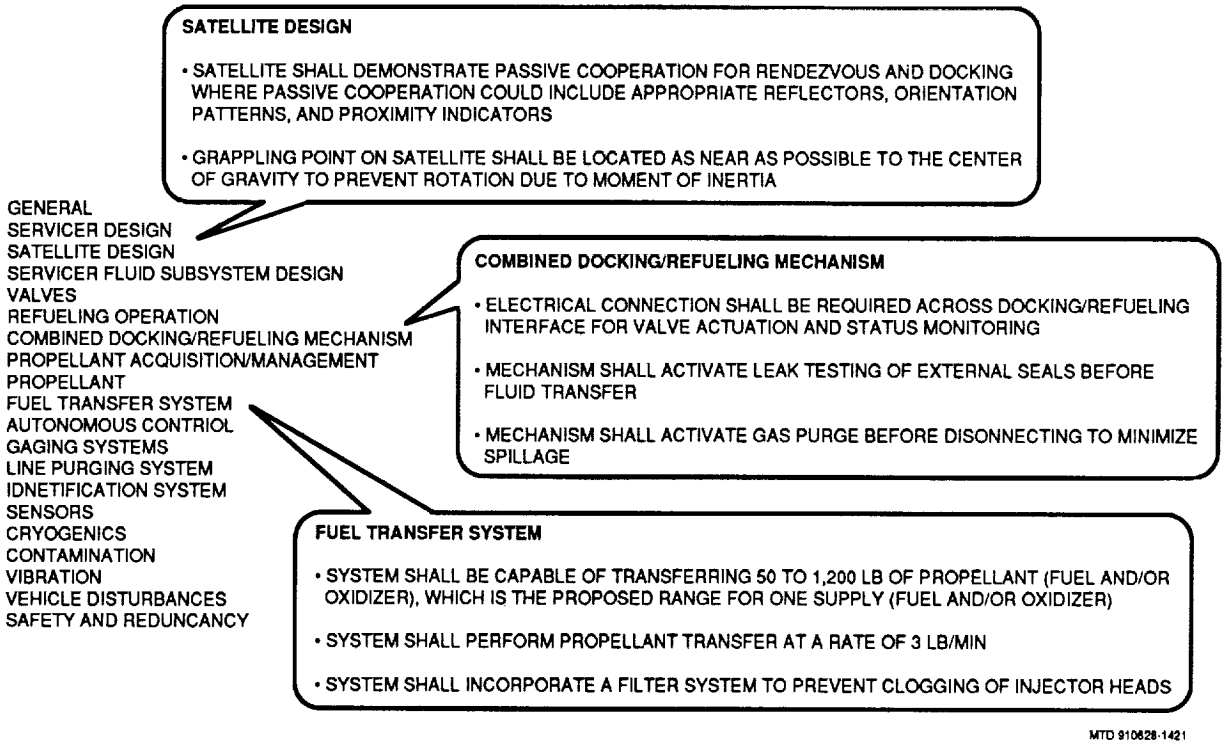
Direct fuel transfer can be accomplished through different methods such as use of an articulated arm to make the connections between fuel lines, electrical connectors and satellite, or a dedicated self guiding umbilical. To identify the best method for making the interface connections, various options were studied. Since the servicer device must dock with the satellite, a feasible approach is the combination of the device for direct fueling and the docking device to form a combined docking and refueling automated umbilical. Unlike an articulated arm that may perform many other tasks, this system is an intelligent mechanical system that only performs one function: docking/mating interface connections. This approach minimizes the interface area/envelop between the servicer and satellites, which reduces the constraint on satellite design. Furthermore, both the docking and mechanism developed for making fluid and electrical connections could utilize the same gross positioning and self-alignment system, which reduces system weight. Both docking and fueling connectors could be driven by the same set of redundant motors. Thus, the chances of misalignment or snagging would be lessened because the interfaces are rigid and follow a predetermined path. Finally, since both docking and making the interface connections are performed in parallel, task operation time is reduced.

AUTOMATED/ROBOTIC REFUELING GUIDELINES

A preliminary set of design guidelines were developed to cover both satellite and servicer subsystems. These guidelines will be used by satellite designers to design the interfaces compatible with the servicer. Guidelines cover the following areas:

- General
- Servicer design
- Satellite design
- Servicer fluid subsystem design
- Refueling operation
- Combined docking and refueling mechanism
- Propellant acquisition/management
- Propellant
- Fuel transfer system
- Autonomous control
- Gaging systems
- Line purging system
- Identification system
- Sensors
- Cryogenics
- Contamination
- Vibration
- Vehicle disturbances
- Safety and redundancy

Figure 3 shows typical examples of these guidelines for depicted areas. Adherence to these guidelines ensures



"Figure 3 - Summary of Guidelines."

compatibility between the satellite and the servicer. The guidelines provide a compatible interface envelop in which one servicer could service different satellites. It is only through such approach that on-orbit refueling could be cost effective.

DESIGN STANDARDS

Design guidelines were used to conceptualize a series of standards (design solution) that could be used in the satellite and servicer design. Where feasible, satellite designers should be encouraged to use standardized designs. This will cut down on development, certification and operation costs. Preliminary design standards where developed for these two areas:

- Servicer fluid transfer system and satellite propulsion system
- Combined docking/refueling automatic umbilical

Servicer Fluid Transfer and Satellite Propulsion System

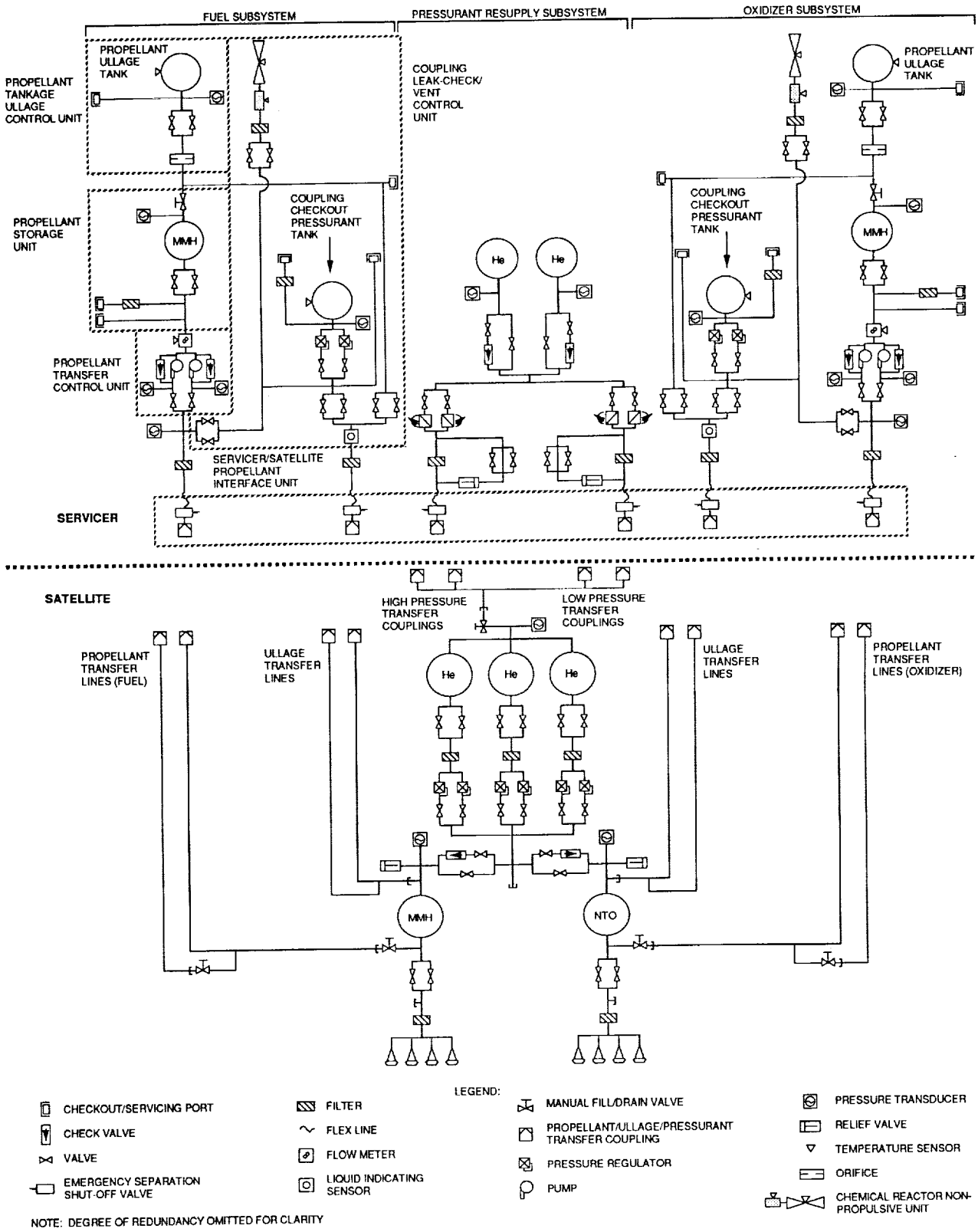
Figure 4 shows the schematic for the servicer fluid/pressurant transfer system and the propulsion system schematic for the satellite. Direct fluid transfer using ullage exchange is recommended as this is a pressure-regulated propulsion system. Here, the degree of redundancy is omitted for clarity. The fuel/oxidizer transfer system, shown in the upper half of the diagram consists of five subsystems:

1. Propellant storage unit
2. Propellant tankage ullage control unit
3. Propellant transfer control unit
4. Coupling leak-check/vent control unit
5. Servicer/satellite propellant interface unit

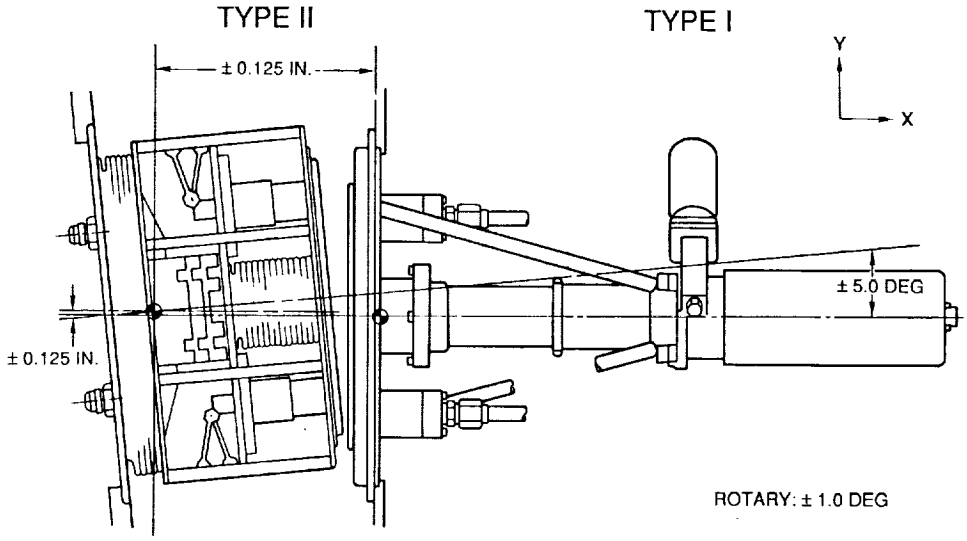
Combined Docking/Refueling Automatic Umbilical

A refueling automatic umbilical should be combined with the docking mechanism for these four reasons: Interfaces are simple; operations are simple since only one mate/demate operation is performed; a set of redundant motors could drive docking and utility connectors; the close proximity of the fluid and electrical connectors to the docking point minimizes mating misalignment. Problems of safety and added complexity preclude the recommendation of a quick-disconnect (QD) mechanism. Should a failure occur during demating, either pyrotechnic or nonpyrotechnic release systems may be used. If connectors/umbilical extend beyond the servicer's thermal blanket, insulation may be needed for thermal control. Multilayer insulation (MLI) or heaters active only during fluid transfer could prevent propellant freezing.

A market survey of available connectors identified the Moog automated umbilical connector (AUC) with its ± 5.0 degree misalignment envelop as a potential design (Figure 5) for fluid, data, and power connectors. The misalignment envelop defines the allowable tracking error for a robotic/automatic servicer or a docking system. Since several connectors need to be mated simultaneously, the possibility exists that one of the connectors may stick in the mating process. A possible solution to this problem involves spring assemblies in the design. Each connector would be mounted on an independent spring assembly. When a connector sticks, its axial motion stops and the spring behind it compresses, allowing the remaining connectors to continue to be mated. With redundant connectors, a failure of this nature would not preclude the propellant/pressurant transfer.



"Figure 4 - Fluid Transfer System and Satellite Propulsion System."



"Figure 5 - Moog AUG."

FUTURE TECHNICAL DEVELOPMENT

On-orbit fueling is a desired capability and some satellites such as GRO have already included the necessary interfaces. The desire will grow and will become a necessity as new NASA and DoD programs are launched. SEI is a good example of such programs.

A comprehensive set of design guidelines and design standards should be developed prior to or in parallel with the design of the first class of assets that require on-orbit fueling. Therefore, work performed here should be continued by NASA, DoD, and technical committees to facilitate on-orbit refueling and make it more effective.

REFERENCES

1. Rose, L.J., "Critical Technologies for Resupply of Spacecraft Propulsion Systems (I)," Martin Marietta, Denver, Aerospace.
2. Gorin, B.F., "Refueling Satellites in Space: The OSCRS Program," Fairchild Space Company, Copyright 1986 Society of Automotive Engineering, Inc.
3. Chandler, F.O., "Orbital Resupply Fluid System Design Issues," Rockwell International, Space Transportation Systems Division.
4. "Orbital Spacecraft Consumables Resupply System (OSCRS) Study," Final Report, Contract NAS9-17585, Martin Marietta, September 1987.
5. "A Study of Fluid Transfer Management in Space," Executive Summary, ERNO, ESTEC Contract No. 6013/84, February 1986.
6. "Special Study No. 10 Serviceable Design for SBI," Final Report, Contract F04701-87-C-0065, Rockwell International, January 23, 1989.
7. "Power System Interface and Umbilical System Study," Contract NAS8-33707, Final Report, Lockheed Missiles and Space Company, July 7, 1980.

N93-11981

OPERATOR VISION AIDS FOR SPACE TELEOPERATION ASSEMBLY AND SERVICING

Thurston L. Brooks
Ilhan Ince
Greg Lee

STX Robotics
4400 Forbes Blvd
Lanham, MD 20706

ABSTRACT

This paper investigates concepts for visual operator aids required for effective telerobotic control. Operator visual aids, as defined here, mean any operational enhancement that improves man-machine control through the visual system. These concepts were derived as part of a study of vision issues for space teleoperation. Extensive literature on teleoperation, robotics, and human factors was surveyed to definitively specify appropriate requirements. This paper presents these visual aids in three general categories of *camera/lighting functions, display enhancements, and operator cues.* In the area of camera/lighting functions concepts are discussed for: (1) automatic end effector or task tracking; (2) novel camera designs; (3) computer-generated virtual camera views; (4) computer assisted camera/lighting placement; and (5) voice control. In the technology area of display aids, concepts are presented for: (1) zone displays, such as imminent collision or indexing limits; (2) predictive displays for temporal and spatial location; (3) stimulus-response reconciliation displays; (4) graphical display of depth cues such as 2D symbolic depth, virtual views, and perspective depth; and (5) view enhancements through image processing and symbolic representations. Finally, operator visual cues (e.g., targets) that help identify size, distance, shape, orientation and location are discussed.

OPERATOR VISION AIDS

This paper outlines a number of operator aids that can improve visual performance during teleoperation. The techniques discussed here are specifically related to improving the process of human vision under remote conditions. We have not dealt with the larger field of graphical display of sensory data (such as force/torque, proximity, etc.) except as this data directly improves visual performance. This paper presents these visual aids in three general categories of camera/lighting functions, display enhancements, and operator cues.

CAMERA AND LIGHTING FUNCTIONS

Camera and lighting functions were found to take a significant portion of an operators time. Operator aids could make a significant performance improvement in this area. Concepts discussed are: (1) automatic end effector or task tracking; (2) novel camera designs; (3) computer-generated virtual camera views; (4) computer assisted camera/lighting placement; and (5) voice control.

Automatic Tracking of End Effector or Task

The function of any automatic tracking aid is to relieve the operator of having to perform additional tasks associated with camera and lighting movement.

Uhrich [1978] reported the first known attempt to track the end effector automatically while performing a task. The camera pan and tilt motions tracked the end effector in only two manipulator joints of motion, but for limited areas of operation the system kept the end effector centered in the camera view. This simple study showed a subjective improvement in operation (actual experiments were not performed). Uhrich reported that although disorientation problems were anticipated, they were surprised to find that the operators could utilize end effector tracking without disorientation.

However, Brooks [1978, Bejczy 1980] found that automatic tracking of end effector motion could create situations in which information normally available to the operator was masked, resulting in operational problems. Specifically, auto-tracking across backgrounds without features or texture gave the operator the impression that the manipulator was not moving, because the end effector remained at the screen center without apparent motion. Under master-slave manipulation, subjects became frustrated due to differences of stimulus-response compatibility (the operator's hand was moving but without apparent effect on the end effector). Additionally, under a non-analogic control such as rate, the

operator did not even realize motion was occurring at all! Brooks' auto-tracking scheme is shown in Figure 1.

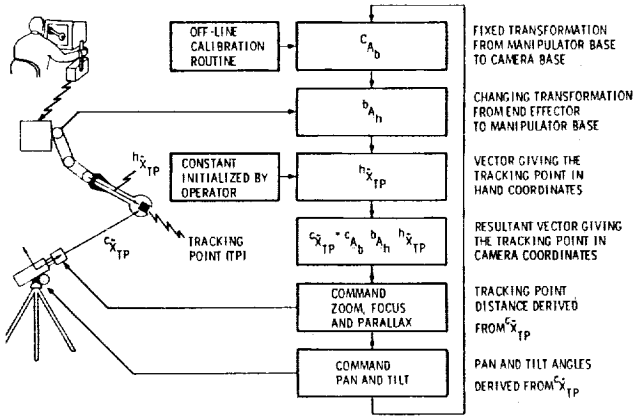


Figure 1: An automatic end effector tracking system can be based entirely on proprioceptive feedback (manipulator joint angles) from the manipulator. The operator simply puts the cross-hairs on the object to be tracked (end effector, tool, point on arm, etc.) and commands the system to maintain that object in the screen center [Brooks 1979, Bejczy 1980].

Another potentially useful operator aid would be to automatically keep the whole manipulator in the field-of-view so that the kinematic configuration could be observed. This concept will be very important as generalized hand controllers and redundant degree-of-freedom arms become more prevalent in teleoperation. Unfortunately, in most task environments, it simply is not possible to simultaneously view the entire arm, and therefore, a graphical 3D representation is believed to be a more practical solution.

In addition to proprioceptive tracking, auto-tracking can also be implemented through machine vision to allow the task motion to be followed [Bejczy 1980]. Using image processing to determine the task/robot relationship, the system could automatically aim the cameras at a point of interest and maintain a fixed relationship of the end effector with the task to relieve the operator of these burdens [Brooks 1979]. Figure 2 illustrates a simple visual method to determine the task-to-telerobot relationship using "labels" that was implemented at JPL [Brooks 1980, 1982 and Bejczy 1980]. Task/object tracking can also involve more sophisticated machine processing such as template-matching or object modeling techniques. Visual servoing and auto-tracking have also been investigated for the OMV, Shuttle umbilical mating and satellite attitude determination [McAnuity 1985, Harrison 1986, Russell 1986, Feddema 1989, KSC 1990 a&b].

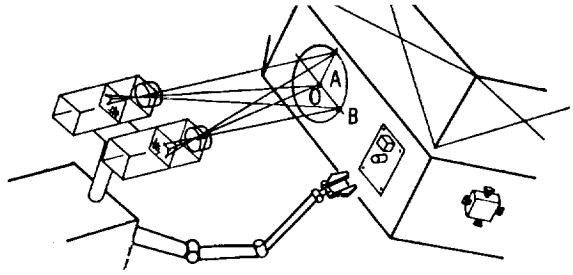


Figure 2: Automatic tracking of task motion can relieve the operator of end-effector station-keeping and camera-aiming duties [Brooks 1980].

New Camera Designs

There are endless possibilities for new and novel camera designs that would improve teleoperation and aid the operator. Two are of interest because they solve recognized problems: glare and acuity.

The first problem is one of specular reflections and glare in the space environment. A specialized camera could help solve this problem through the use of liquid crystal technology. In essence, a CCD camera could have a liquid crystal shade (LCS) with identical resolution placed over the CCD array. Through image processing the system could determine the pixels receiving to much light and the LCS in those areas could suppress or block the offending rays (see Figure 3).

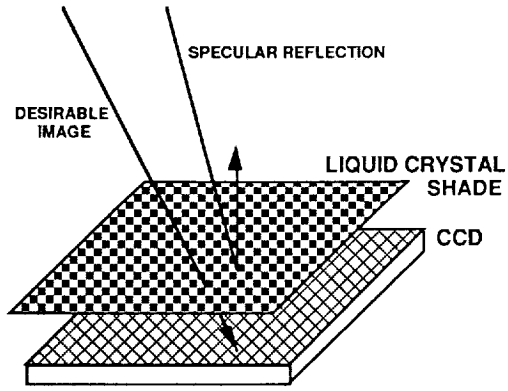


Figure 3: Glare and specular reflection could be removed from images through the use of a liquid crystal shade.

The development of a camera with a foveal view of very high resolution and a surrounding peripheral view of low resolution is another novel camera concept first suggest by Carl Ruoff at JPL.[1] Designed properly, a single CCD chip could pack a 1-2° foveal center and greater than a 100° peripheral view on one single NTSC channel (Figure 4). A design like this would permit high resolution and peripheral view without a bandwidth increase.

[1] Discussion between T. Brooks and Carl Ruoff at JPL in 1979.

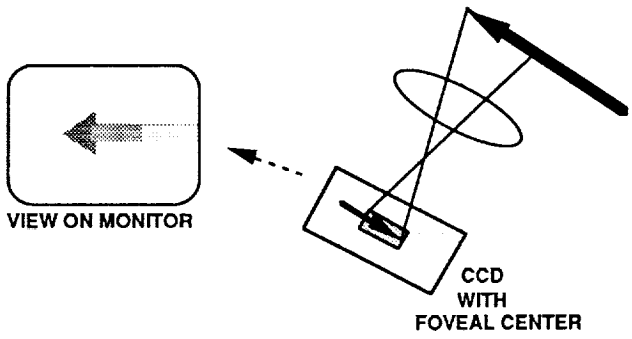


Figure 4: A CCD chip with a high-resolution fovea and a low resolution peripheral could permit the use of a low NTSC bandwidth with high-acuity.

Preview Scenes

Synthetic camera image generation based on CAD models of the task environment could permit the astronaut to determine if a particular view would meet task performance requirements before committing the resources. These preview scenes give the astronaut "what-if" capability to explore alternative solutions.

Computer Assisted Camera/Lighting Placement

A graphical representation of current state of cameras and lighting could be a useful overview display. A display such as the one shown in Figure 5 would permit the astronaut to quickly determine which camera(s) and light(s) would result in the optimal viewing conditions for a task. The display could present an overhead view illustrating light beams and camera field-of-view based on current or proposed zoom, focus, convergence, light intensity, etc. If a method for entering up-to-date orbit status was available, the system could also show Sun illumination, including reflected light. A perspective view showing expected illumination and visibility from a selected, or desired, camera could also be generated and displayed by simply selecting the desired camera vantage point (Figure 6).

Another option would be to have computer-aided camera/lighting optimization, wherein the computer generates a graphic suggesting the best camera and lighting position for a specified task location. For example, after positioning the FTS, the astronaut could request the system to optimize lighting and camera views for current orbit trajectories. With advanced computer aiding, the astronaut could simply "point-and-go", leaving the system to select the best camera, set focus, zoom, and convergence, and adjust light levels automatically. Other possibilities include: (1) having the camera system automatically maintain a fixed image size of an object-of-interest on the monitor, (2) allowing the astronaut to specify a desired depth of field, and (3) automatic contrast adjustment through optimization of lighting and camera adjustments.

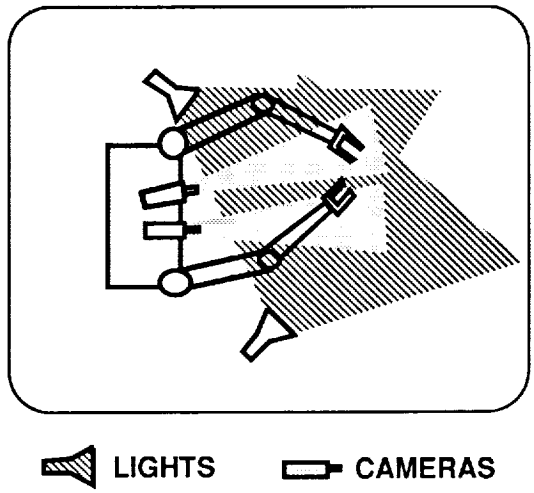


Figure 5: A graphical display of lighting/camera pointing angles and FOV could provide status-at-a-glance.

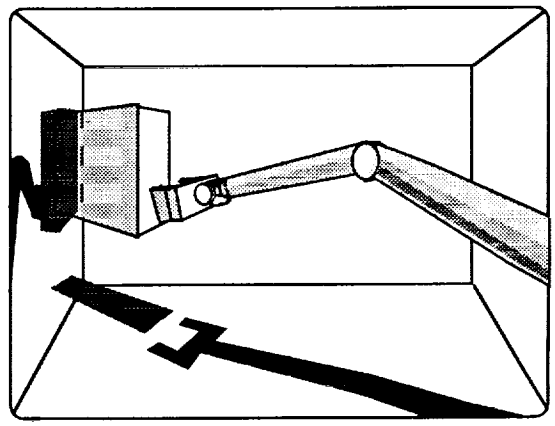


Figure 6: A computer-generated perspective view could show predicted visibility from a selected camera location.

As a final note of interest, robotic cameras have been used in the broadcast industry for a couple of years now, with reported benefits of smoother panning motions, allowing observers to more easily follow details and maintain perspective [Lehtinen 1990].

Voice Control

Voice control allows the operator to simultaneously maintain hand controller movements, observe CCTV monitors, and issue verbal commands. Bejczy [1981] speculates that voice control does not disturb the operator's visual attention or manual control functions, and that it minimizes mental distractions. However, early experiments with voice control in the Manipulator Development Facility at JSC, although deemed successful, indirectly highlighted an important aspect of voice control — its discrete

nature. Voice control is a discrete system input rather than an analog one, and hence, it is best for discrete functions. Voice is quick and efficient for switching a camera to a monitor (e.g., "Camera B3 on Monitor 1"), or homing in on a known location (e.g., "Zoom in on ORU Handle"), but it is not as efficient for panning and tilting functions which results in a number of overshoots or slow motions (e.g., "Pan Left...Faster... Slow...Stop...Pan Right...Slow...Stop").

At NASA, a number of voice control systems have been developed and tested for controlling cameras and lighting. For example, at the Jet Propulsion Laboratories a teleoperator control station has been developed in which cameras and monitors can be selected, zoomed, focused, panned, and tilted by voice control. Johnson Space Center's voice system has been flight tested onboard the Shuttle with some success [Foley 1991]. The JSC voice control system allowed complete hands-free control of CCTV functions:

1) monitor selection
2) camera selection
3) pan, tilt, focus, iris, zoom, and scene track

DISPLAY ENHANCEMENTS

A number of operator display enhancements were discovered to be potentially useful: (1) zone displays, such as imminent collision or indexing limits; (2) predictive displays for temporal and spatial location; (3) stimulus-response reconciliation displays; (4) graphical display of depth cues such as 2D symbolic depth, virtual views, and perspective depth; and (5) view enhancements through image processing and symbolic representations.

Zone Displays

A zone display highlights areas of the visual field that place restrictions on the operator for one reason or another. Two of the more obvious possibilities for restricted access are areas where collisions are likely to occur and locations that the operator cannot reach due to limits of the arm geometry (i.e., workspace). Figure 7, for example, illustrates a workspace restriction display.

Another useful display is one that highlights imminent collisions of the telerobot that can occur with its environment, its carrier (e.g., the RMS), or with itself. It is preferable to prevent all undesirable collisions, but a complete lockout of all contact would render the teleoperator incapable of performing its mission. A display that could warn the astronaut of a collision before it occurs would give the astronaut the option of taking corrective action or proceeding if essential to the mission. An imminent collision display could operate in a number of different modes:

1) The offending object(s) could be "painted" yellow to indicate an impending collision and red to indicate that a collision had occurred.

2) Stick figures of imminent objects could be superimposed on the real-time video to indicate collision status [Crane 1986].

3) All possible collision points in the camera's view could be false colored (e.g., yellow) to indicate that a collision is possible. This is likely to be quite a "busy" display which could be more confusing than helpful.

4) A graphic simulation of the remote manipulator could be false colored in the general area where a collision is expected as shown in Figure 8.

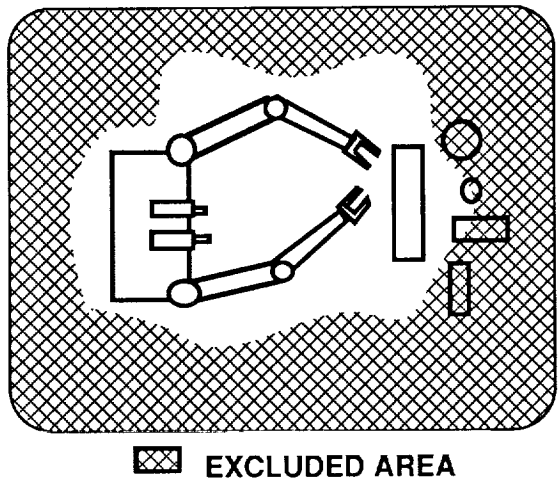


Figure 7: A graphical display of workspace constraints could provide the crew important information about whether a task could be completed in the current position.

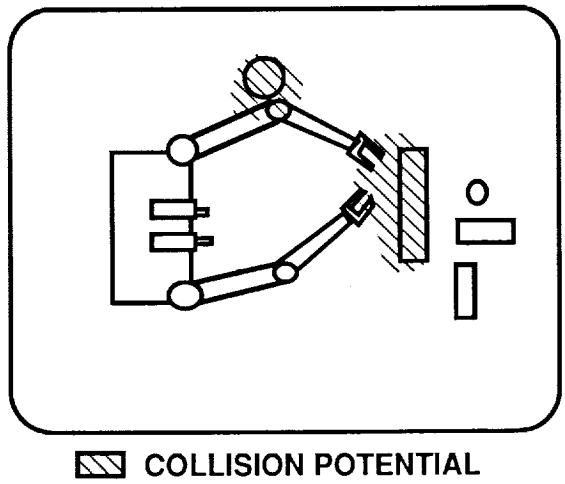


Figure 8: Graphical display of potential collision zones could prevent damaging contact while allowing the operator the option of deliberate contact.

5) A graphic overview of the task environment and remote system could highlight the highest likelihood collision point based on calculated approach velocity and separation distances. This collision display highlights the highest probability collision, not necessarily the closest points. For example, if the arm is moving away from an object it would be impossible to collide with the object even if one were nearly touching it.

Another potentially useful display would highlight the instantaneous workspace of the slave manipulator due to the limits of the master controller's motion (i.e., limits due to indexing of the master when there is not a one-to-one correspondence between master and slave). Indexing is a control strategy in which the operator can re-reference the master hand controller relative to the slave to maintain the hand controller and its movements within an optimum volume. Typically, indexing is done to allow the operator to use the hand controller in a small volume of space while controlling a remote manipulator in a larger volume without resorting to scaling. Unfortunately, once the slave is in position and the hand controller is re-referenced, there is no guarantee that the current location has sufficient freedom to permit the task to be performed without re-indexing in the middle of the task. This can be a nuisance, if not a detriment, if the task requires the operator to index just to move the manipulator a few additional millimeters. A solution to this problem would be to display the achievable slave workspace as a function of the indexed master movements as shown in Figure 9. An index-workspace display such as shown in Figure 9 could assist the operator in placing the slave in an optimal position for completing a task with a minimum of re-referencing actions.

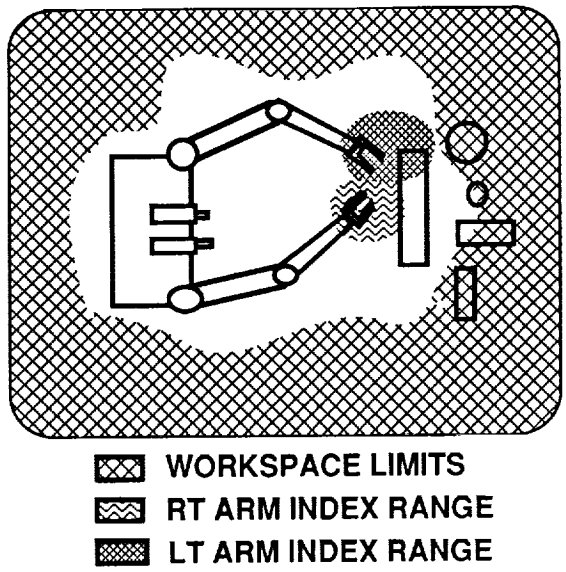


Figure 9: Display of indexed workspace could allow the operator to determine whether a task could be completed in the current indexed position, or whether the arms must be repositioned to complete the task.

Predictive Displays

Predictive displays, as used here, are displays that estimate where objects are or will be. When we speak of where objects *are (or should be) at this instant*, we use the term spatial predictive displays. When we speak of where objects *will be in the future*, we use the term temporal predictive displays.

Figure 10 is an example of a spatially predictive display in which a CAD model is superimposed on the actual video image for the purpose of highlighting the location of components when they cannot be seen due to obscuration or insufficient lighting.

Spatially predictive displays can also be used to indicate where the telerobot "thinks" an object is, versus where its true location is. GE has used this technique to allow an operator to correlate a CAD model to the real world and bring the two into alignment [Oxenberg 1988]. In the GE system used at JPL, the operator updates the database by superimposing a wireframe line drawing of an object on a video scene containing the object. The real-world and model are brought into alignment by having the operator match vertices of the model to their equivalent vertices on the image.

Overlay of stick figures on video can also be used to compare planned (computer) versus actual performance. Russell [1986], for example, has suggested superimposing stick figures on live video to aid in satellite docking.

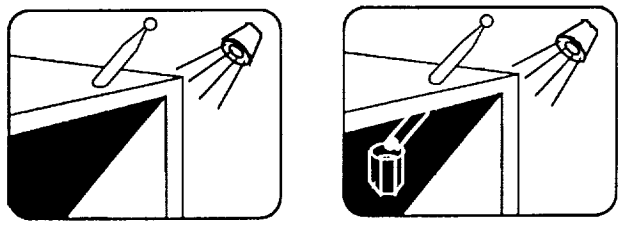


Figure 10: Spatially predictive displays superimpose known data from CAD models or sensors on the real-time video to provide the operator information that is not otherwise readily available.

Temporal predictive displays project what will happen at some point of time in the future (Figure 11). Sheridan [1986] first suggested using computer generated images to predict robot response to operator commands under a time delay. Since then, the concept has developed to

include full dynamic models in which the operator interacts as if actually manipulating the remote object. The operator's position and force trajectories are then fed to the remote robot which performs the task using the human generated trajectories as controller inputs. Reported performance improvements have been significant [Sheridan 1987, 1989].

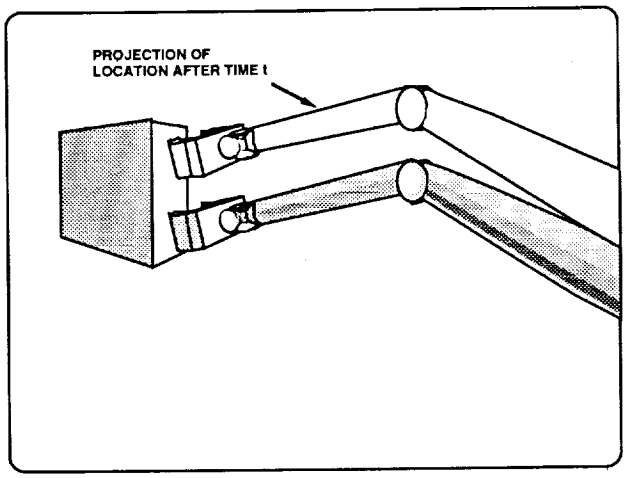


Figure 11: Temporally predictive displays project what will happen in the future based on controller actions occurring now.

Stimulus-Response Reconciliation

A well-recognized but perplexing problem, that anyone who has operated a remote manipulator has experienced, is control reversal, resulting in unexpected movement. Control reversal (also known as cross-coupling, stimulus-response mismatch, and orientation-display incompatibility) has been reported in fly-by-wire applications [e.g., Van Cott 1972], teleoperated vehicles [e.g., McGovern 1988], and telemanipulators [e.g., Brooks 1979, Smith 1988].

Brooks [1979] developed a classification of stimulus-response mismatch in an effort to understand the problems observed during experiments. He concluded that cross-coupling was due to mechanical, geometrical, or observational stimulus-response (SR) mismatch. See Figures 12-14 for illustrations of these conditions. The key to understanding SR mismatch is to recognize that it is caused by a disparity between what the operator expects and what s/he observes.

The effects of operator stimulus-response incompatibility have been recognized for many years, but have only recently been receiving research attention. Smith [1990] concludes that a workable approach may be to use "computer-mediated transformation of video images of the telemanipulator to a spatially compliant form before they are displayed to the operator." This solution is currently only possible in a 3D graphical format, but increasing processor speeds will make real-time image manipulation a viable alternative in coming years.

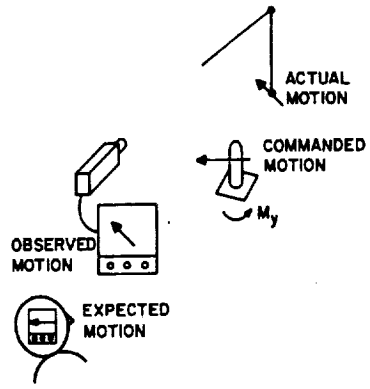


Figure 12: *Mechanical Stimulus-Response Mismatch.* The operator desires to move to the left, but mechanical cross-coupling in the hand controller causes the manipulator to move differently. The observed and actual motion are the same, but incorrect [Brooks 1979].

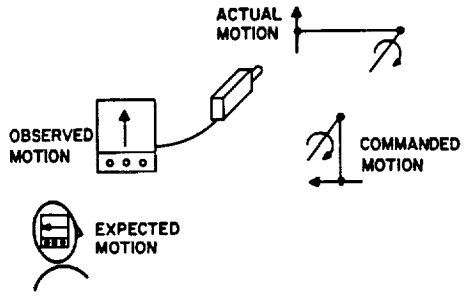


Figure 13: *Geometrical Stimulus-Response Mismatch.* The operator commands a move to the left, but dissimilar geometric relationships between the hand controller and telerobot causes the manipulator to move differently. The observed and actual motion are the same, but not what the operator expected [Brooks 1979].

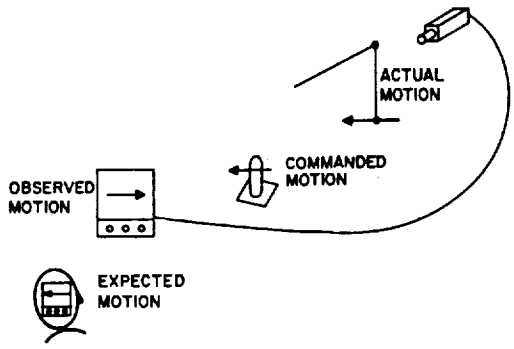


Figure 14: *Observational Stimulus-Response Mismatch.* The operator commands a move to the left and the manipulator moves as commanded, but due to the positioning of the camera the observed motion is different than the expected motion [Brooks 1979].

One possible solution, suggested by Crane [1989], is to use screen coordinates as the control frame. The idea being that regardless of manipulator orientation or position, operator commands will be referenced with respect to the screen coordinates that he is currently viewing. Hence, a camera sighted along the end effector axis would move forward whenever the operator pushed the control stick forward in the screen direction. This is equivalent to using an end effector control frame, provided that the camera and end effector have a rigid connection. The true difference between screen coordinates and end effector control becomes clear when one has a camera located at least one or more joints from the end effector. To illustrate (see Figure 15), imagine using an end effector control frame with the camera fixed at the manipulator base, and rotate the end effector 180° about the vertical axis so that the end effector is facing the camera. Now, using an end effector control frame, rightward stick motions cause the end effector to move left in the image. If control were in screen coordinates, a right stick motion would cause a rightward end effector motion. It can be hypothesized that this form of stimulus-response remediation could have disastrous effects when more than one monitor (screen control frame) is used if the operator retains the mental model/relationships from the previous screen. Regardless of this possible shortcoming, this scheme does warrant investigation in a teleoperated setting.

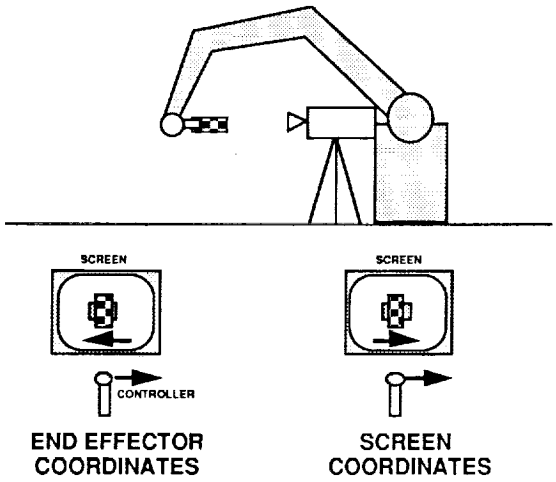


Figure 15: Stimulus-response incompatibility results in operator confusion. A suggested solution is to use a screen based control frame so that hand controller movements and screen movements correspond.

Display of Depth thru Graphics

Depth information is fundamental to most teleoperated tasks. However, for many teleoperation tasks, either a needed view will not be available, or available views will not provide appropriate visual depth cues. Fortunately,

2D Symbolic Depth

The first method is a symbolic representation of various depth cues which are superimposed on the real-time image. An example of such a display is shown in Figure 16. The cross in the middle represents the pitch and yaw of the end effector, while the two converging parallel lines represent range to the task. The astronaut knows that the end effector is properly aligned when the yaw and pitch rectangles are centered on the reticule, and the converging range rectangles decrease to zero. The data for this display can be generated from either a CAD database, proximity sensors [Bejczy 1980], or laser range finders.

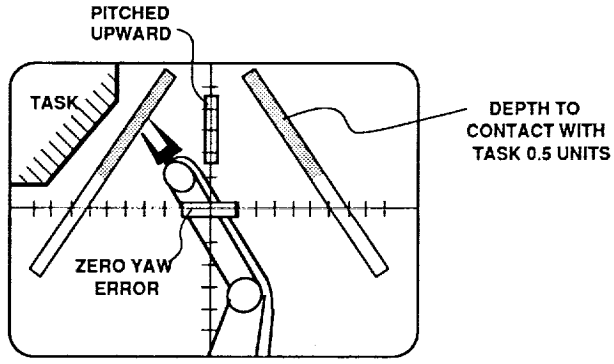


Figure 16: A 2D symbolic depth and alignment display for improving end effector control with only a single camera view.

Virtual Views

A second method of providing depth information is to generate a simulated view as if seen from a virtual camera placed at a useful location near the task (see Figure 17). Through virtual scene generation, the astronaut could see the task/manipulator interface from any angle/location desired, even from within the object looking out if that would aid performance. Obviously, a view orthogonal to the existing video camera view would provide the most accurate information for motions along the camera axis.

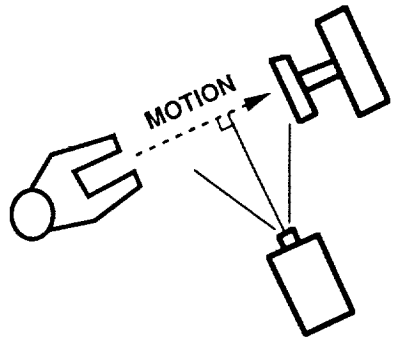


Figure 17: Virtual camera views can be graphically generated from any desired vantage point. The motion of the teleoperator determines an optimal orthogonal camera view.

However, other viewing angles would also allow the astronaut to "fly" around the task to observe progress from all angles.

Perspective Depth

A third method of displaying useful depth information is through the use of perspective grids [Stark 1987, Kim 1987, and Wickens 1990). As shown in Figure 18, a perspective grid is superimposed on the image, and markers are dropped to the grid to indicate the control point position to the operator. The goal state could also be indicated through the use of some appropriate symbol. Experiments by Kim [1987] have shown that a perspective grid can improve human performance.

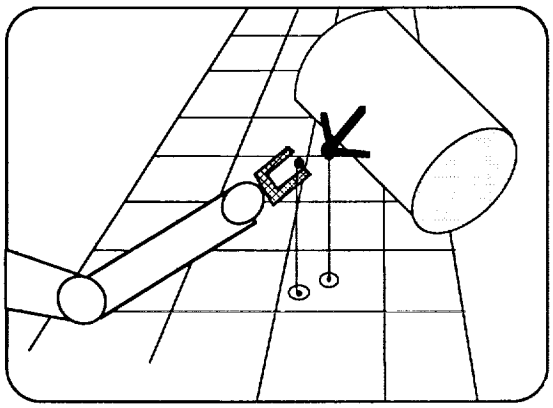


Figure 18: A perspective grid display superimposed on a single camera view to aid the operator.

View Enhancements

Through the use of image enhancement techniques, it is possible to aid the astronaut in performance of tasks which might be impossible without such aids. For example, a low-contrast picture could be "stretched" in real-time to bring out details that would otherwise be unobservable [Gonzalez 1977]. Or features within shadows could be expanded to effectively increase the dynamic range of the display. Another possibility would be to delineate edges or surfaces which might be indiscernible due to similar texture or surface reflectivities.

Another enhancement might involve deliberate image distortions to aid in grasping/docking operations. For example, images could be plastically "stretched" in screen dimensions which highlight errors. Yet another enhancement would be to remove image blurring due to task motion.

Image enhancement could also improve specular reflections through dynamic memory of prior scenes. As the sun reflects off thermal blankets during orbit, specular image portions could be removed and lost detail could be filled in by computer memory (acquired when the Sun was in a different position) or computer generated from a CAD data base.

Symbolic enhancements are artificial graphic elements which aid the operator by providing information that might otherwise not readily be available. One such enhancement might be to turn the end effector different colors to indicate gripper **status** directly to the astronaut rather than through a secondary display, such as a light on the console. For example, a red gripper would mean the object is not firmly grasped, whereas a green gripper would indicate positive capture. Another enhancement would be to indicate **problem areas** with symbolic colors. For example, a temperature probe in the end effector could be used in conjunction with an ORU CAD data base to indicate problem areas by "painting" a cryogenic cold-spot blue or a hot-spot red (see Figure 19). Symbolic enhancements such as these can be achieved through graphic overlays of the real-time video.

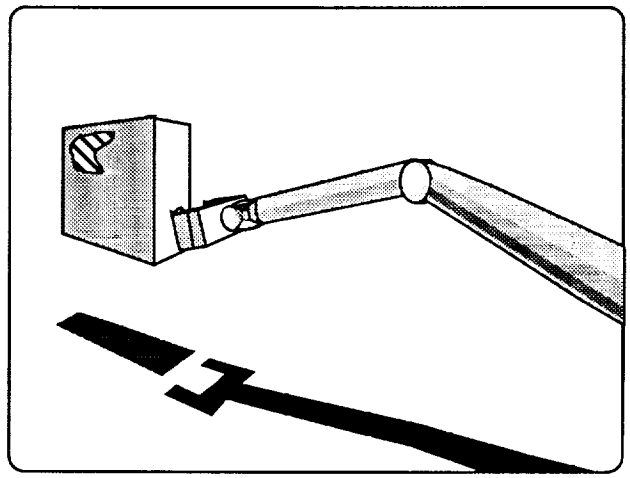


Figure 19: Symbolic enhancements are artificial graphic overlays that supply information thru direct visual analogy. Here, for example, a hot-spot is highlighted to indicate a fault location.

OPERATOR CUES

Ancther important operator visual aid is represented by a class of visual "cues" used to improve proper orientation and positioning of the end effector when viewing conditions are limited. Figure 20 illustrates a visual cue used for grasping payloads with the Shuttle RMS. These types of cues are typically constructed as an integral part of the payload in a position that can be directly viewed from an end effector mounted camera. Cues help identify size, distance, shape, orientation and location of objects and aid task performance. Cues minimize task time and insure proper execution by providing a pre-existing, logical relationship between end effector, task and operator actions.

Cues typically use high-contrast backgrounds and foreground elements to insure they are readily discernable. In color systems, colors of differing hues can be used. Alignment cues typically use single or multiple stripes or patterned markings. Range cues typically use size on the screen or comparison markings (e.g., the end effector is at the proper range when a marking on the end effector and a marking on the task are the same length).

Another method of visual enhancement for grasping or docking functions would be to generate graphic cues superimposed on the monitor. For example, alignment of the real-time image with a graphic outline or "stick-figure" could indicate successful operation. Figure 21 is a concept in which moire circles are superimposed on the end effector video image to indicate yaw and pitch alignment. Graphic overlay techniques would allow tasks to be performed without the need for mounting or painting special target cues on the satellite.

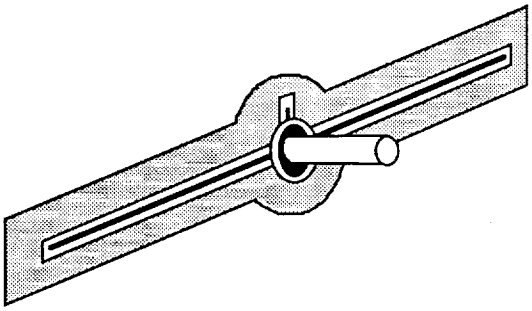


Figure 20: Target cues used by the RMS to achieve accurate alignment.

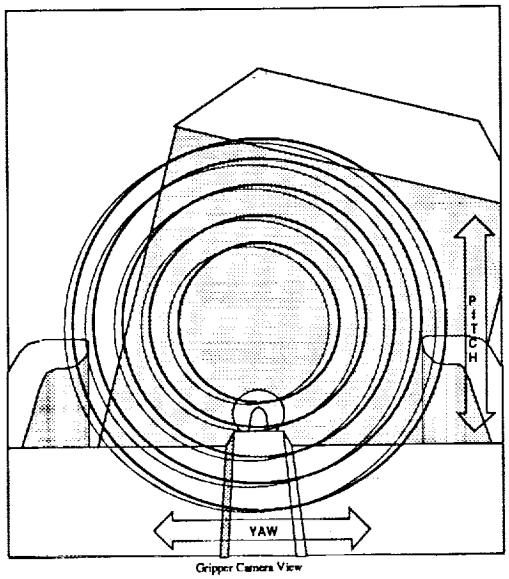


Figure 21: Moire circles superimposed on the real-time video image indicates yaw and pitch alignment.

CONCLUSIONS

This paper has presented a number of different concepts for improving operator performance through visual aids. Operator vision aids were broken down into three primary categories: (1) remote camera and lighting functions, (2) display aids, and (3) operator cues.

In the area of remote camera and lighting, we presented concepts for:

- **Automatic tracking of the end effector or task with the camera and lighting.**
- **New camera designs.**
- **Simulated camera views to aid in proper camera and lighting placement.**
- **Computer assisted camera and lighting placement.**
- **Voice control of camera and lighting functions.**

In the area of display aids for the operator, we presented concepts for:

- **Displaying restricted zones such as collision and object limits.**
- **Predictive displays that help the operator identify where something is or will be after the control action is taken.**
- **Displays that reconcile visual feedback with operator inputs to prevent display/control mismatch.**
- **Methods for providing the operator with depth information through graphical aids.**
- **Enhancement techniques that provide the operator information through image modification and symbolic image manipulation.**

Finally, *visual cues* can used to improve proper orientation and positioning of the end effector when viewing conditions are limited. Cues help identify size, distance, shape, orientation and location of objects, and aid task performance. Cues minimize task time and insure proper execution by providing a pre-existing, logical relationship between end effector, task, and operator actions.

It is hoped that this report condenses and focuses both the current state-of-the-art and the future challenges of applying operator visual aids to teleoperation.

REFERENCES

Bejczy, A., and Brooks, T., "Advanced Control Techniques for Teleoperation in Earth Orbit," 7th Annual Symposium of the Association for Unmanned Vehicle Systems, June 1980.

Bejczy, A. , Dotson, R., Brown, J., and Lewis, J., "Voice Control of the Space Shuttle Video System," 17th Annual Conference on Manual Control, June 1981.

Brooks, T., Discussion with Paul Heckman of NOSC on Uhrich and Heckman's end effector tracking results. These discussions ultimately led to the development of a 6 DOF end effector tracker at JPL in 1980 in Tony Bejczy's lab. The results were unpublished since statistical experiments were not performed; however, the tracking algorithms were published in Brooks [1979] and Bejczy [1980], 1978.

Brooks, T., "Superman: A System for Supervisory Manipulation and the Study of Human/Computer Interactions", MIT Man-Machine Systems Laboratory, SEA Grant 04-7-158-44079, 1979.

Brooks, T., "Supervisory Manipulation Based on the Concepts of Absolute vs. Relative and Fixed vs. Moving Tasks," Advances in Computer Technology, Vol. 1, 1980.

Brooks, T. and Cunningham, R., "Satellite Tracking and Retriaval using Stereo Vision," JPL Video Tape, 1982.

Brooks, T. , Ince, I., and Lee, G., "Vision Issues for Teleoperation Assembly and Servicing (VISTAS): A survey of Visual requirements for Teleoperation by the FTS," STX Report, No. STX/ROB/91-01, January 1991.

Crane, C. and Duffy, J., "An Interactive Animated Display of Man-Controlled and Autonomous Robots," Proceedings of the ASME Computers in Engineering Conference, Chicago, 1986.

Crane, C., et.al., "Telepresence System Development for Application to the Control of Remote Robotic Systems," Proceedings of the NASA Conference on Space Telerobotics, Vol. III, Jan. 1989.

Feddema, J., Lee, C., and O. Mitchell, "Weighted Feature Selection Criteria for Visual Servoing of a Telerobot," Proceedings of the NASA Conference on Space Telerobotics, Vol. III, Jan. 1989.

Foley, Tico. Telephone Interview by Greg Lee with Tico Foley of Johnson Space Center, January 31, 1991.

Gonzalez, R., and Wintz, P., Digital Image Processing, Addison-Wesley, 1977.

Harrison, F., and Pennington, J., "System Architecture for Telerobotic Servicing and Assembly Tasks," SPIE Vol 729 Space Station Automation II, 1986.

Kim, W., et.al., "Quantitative Evaluation of Perspective and Stereoscopic Displays in Three-Axis Manual Tracking Tasks," IEEE Transactions on Sytems, Man and Cybernetics, Vol. SMC-17, 1987 (a).

Kim, W., Tendick, F., and Stark, L., "Visual Enhancements in Pick-and-Place Tasks: Human Operators Controlling a Simulated Cylindrical Manipulator," IEEE Journal of Robotics and Automation, RA-3, 1987 (b).

KSC, "Robotic Target-Tracking Subsystem," NASA Tech Briefs KSC-11447, 1990(a).

KSC, "Research and Technology 1990 Annual Report," NASA Technical Memorandum 103811, Kennedy Space Center, 1990(b).

Lehtinen, R., "Corporate Camera Robotics," Video Systems Magazine, September 1990.

McAnulty, M., and Vinz, F., "Machine Vision and the OMV," NASA/ASEE Summer Faculty Research Fellowship Program MSFC, NASA-NGT-01-008-021, Aug. 1985.

McGovern, D., "Human Interfaces in Remote Driving", IGC 7th Conference on Real-Time Stereoscopic Video Display Systems for Telerobotics, CAD/CAM and Medical Imaging, 1988.

Oxenberg, S., Landell, B., and E. Kan, "Geometric Database Maintenance Using CCTV Cameras and Overlay Graphics," SPIE Vol. 1006 Space Station Automation IV, 1988.

Russell, R. and D'Arcy, A., "Video-Based Satellite Attitude Determination," SPIE Vol. 729 Space Station Automation II, 1986.

Sheridan, T. and Verplank, W., "Human and Computer Control of Undersea Teleoperators," Man-Machine Systems Lab, MIT, ONR Report, July 1978.

Sheridan, T., "Human Supervisory Control of Robot Systems," IEEE Robotics and Automation, Vol. 2, 1986.

Sheridan, T., Kruser, D., and Deutsch, S. (Eds), Human Factors in Automated and Robotic Space Systems, Proceedings of a Symposium, National Academy of Sciences - NRC, National Academy Press, Washington, DC, 1987.

Sheridan, T., "Telerobotics," Automatica, 25(4), 1989.

Smith, R, "Human Visual Requirements For Control and Monitoring of a Space Telerobot", Ergonomics of Hybrid Automated Systems I, Elsevier Science Publishers, 1988.

Smith, T., Stuart, M., Smith, R., and Smith, K., "Interactive Performance Variability in Teleoperation: Nature and Causes", Ergonomics of Hybrid Automated Systems II, 1990.

Stark, L., et. al., "Telerobotics: Display, Control, and Communication Problems," IEEE Robotics and Automation, 1989.

Uhrich, R.W. and Fugitt, R. B., "Investigation of Control and Viewing Techniques for Use with Remote Undersea Manipulators", NOSC TR 233, 1978.

Van Cott, H. and Kinkade, R., Human Engineering Guide to Equipment Design, US Government Printing Office, 1972.

Wickens, C.,"Three-Dimensional Stereoscopic Display Implementation: Guidelines Derived from Human Visual Capabilities," SPIE Vol. 1256 Stereoscopic Displays and Applications, 1990.

N93-11982

A SMART END-EFFECTOR FOR ASSEMBLY OF SPACE TRUSS STRUCTURES

William R. Doggett; NASA Langley Research Center; MS 152D;Hampton Va 23665-5225
Marvin D. Rhodes; NASA Langley Research Center; MS 199; Hampton Va 23665-5225
Marion A. Wise; Lockheed Engineering and Sciences Company; Hampton Va 23665-5225
Maurice F. Armistead; Planning Research Corporation; Hampton Va 23665-5225

ABSTRACT

A unique facility, the Automated Structures Research Laboratory, is being used to investigate robotic assembly of truss structures. A special-purpose end-effector is used to assemble structural elements into an eight meter diameter structure. To expand the capabilities of the facility to include construction of structures with curved surfaces from straight structural elements of different lengths, a new end-effector has been designed and fabricated. This end-effector contains an integrated microprocessor to monitor actuator operations through sensor feedback. This paper provides an overview of the automated assembly tasks required by this end-effector and a description of the new end-effector's hardware and control software.

INTRODUCTION

NASA Langley Research Center has initiated a research program to develop methodologies for automated in-space assembly of large truss structures. A laboratory test facility, called the Automated Structures Assembly Laboratory, has been recently developed in which a regular tetrahedral truss structure, that includes 102 truss members, has been assembled using a specialized end-effector mounted on an industrial robot arm. A description of the truss is included in reference 1 and a description of the facility is included in reference 2. The end-effector currently used in the test facility was designed for, and is restricted to, operations on fixed length struts. Fixed length struts restrict operations to the assembly of structures composed of regular tetrahedral subelements similar to the current test structure. To address the assembly of other structures, such as support trusses for antennas that have a parabolic shape and require members of many different lengths, and to expand the capability of the system, to perform tasks such as the installation of payloads, a new end-effector is required. Also, for the current assembly experiment, the computational requirements have become very large and a distributed computational capability that incorporates dedicated microprocessors to command actuators and verify successful operations through the interrogations of sensors is needed. Therefore, an end-effector which installs one end of a strut at a time and uses a dedicated on-board microprocessor for sensor interrogation and actuator control, was designed and fabricated. This new end-effector will permit assembly of a variety of structures, installation of payloads with a standardized connection, and will provide the first opportunity for concurrent operations within the test facility. The time required to assemble a structure with the new end-effector is expected to increase over that required by the current end-effector, because the new end-effector operates on one end of a strut at a time and is required to capture most struts in a cantilevered configuration.

The purpose of this paper is to describe the new end-effector and the operations involved in truss assembly, describe the relationship between the end-effector and the overall hierarchical control scheme used in the assembly facility, and detail the operation of the end-effector's control software; highlighting the advantages that can be achieved through embedded microprocessor control. Emphasis will be placed on the unique features of the software design. Planned tests and further work will also be discussed.

STRUCTURAL ASSEMBLY TEST FACILITY

The current automated assembly test facility at the Langley Research Center is shown in the photograph of figure 1. Figure 1a is a schematic of the assembly system with the subcomponents labeled and figure 1b is a photograph of the actual assembly laboratory. The facility is a ground-based research tool to permit the development and evaluation of assembly hardware concepts, construction techniques, software; and operator interface systems that are likely to be required for on-orbit assembly operations. The robot arm is an electrically controlled, six degree-of-freedom industrial model that was commercially available when the program was initiated. The arm was selected for the laboratory test operations primarily because of its payload capacity, reach envelope, and positioning repeatability. No modifications to the robot other than those that are commercially available from the manufacturer have been made. The robot is mounted on an x-y Cartesian motion base that provides translational travel to position the base of the robot anywhere in the support track area to within 0.002 of an inch. The truss is mounted and assembled on a rotating motion base at the end of the translational base. Both of the motion bases are designed to minimize positional errors that may be induced by static deformations from the mass of the robot, unbalanced asymmetric truss assembly, and forces exerted by the robot during assembly.

The truss selected for assembly test operations is a regular tetrahedral truss composed of 102 strut members each approximately two meters long. This size and configuration were chosen because they are representative of the support structures that are required for a number of planned and/or proposed missions. The truss also has a number of geometric characteristics that make it desirable for an automated assembly test bed. These characteristics are described in reference 3. The truss members are connected by specially designed connector joints located near the nodes. Each node must be capable of connecting nine members, six that are in the plane of the top or bottom face and three core members that attach the top and bottom faces. The joints are located as close to the nodes as physically practical to minimize the packaging volume of the members for launch. The confined region near the node, however, somewhat complicates assembly operations since the end-effector must have a small size to be capable of installing a member. The truss joint connector is shown in figure 2. The joint is composed of two parts; a connector section that is bonded to the strut tube, and a receptacle section that is mechanically attached to the node. The joint has a ramped entry way to aid in inserting the connector into the receptacle. After insertion, the locking nut is turned to draw the connector plunger into the receptacle pocket and to preload the joint to eliminate free-play in the truss, thus providing a structurally predictable assembly unit.

The end-effector shown in figure 1 is the first generation model developed especially to assemble the truss shown. The initial tests were developed around the use of this specialized tool to establish a test database and provide experience for the development of the operator interface while the facility hardware was coordinated and tested.

The facility is directed and controlled by several digital computers that are currently serially connected through RS 232 communication lines to transfer commands and status information as shown in figure 3. For the current system all end-effector operations are controlled by the robot computer. This configuration was adopted because the robot computer was capable of directly monitoring end-effector sensors and executing commands through integrated analog-to-digitial and digital-to-analog converters. The status of the assembled structure, location of all truss members, position of the robot arm and motion bases, and status of the end-effector are all stored in appropriate databases. At the highest level, the operator selects the desired function to be performed from a menu of permissible commands. The executive program checks the current status of all components and the status of the assembled structure and determines if the requested command is permissible. All assembly operations are performed under the supervision of the computer executive, which operates as an expert system. During the assembly process, a display adjacent to one of the menus advises the operator of commands initiated and their completion status.

Shown in figure 4 is the assembly software structure which is divided into five hierarchical levels: planning, truss element, device, component and verification. The figure indicates typical operator commands at each level, which are decomposed by the automated system into a series of commands for the next lower level. The highest, or planning level, which is not currently implemented, would be driven by an automated task sequence planning tool. A command to "build" a structure or substructure would be decomposed by the planner into a sequence of commands for the second level. To fetch and install a strut requires action by 3 separate devices; the motion bases, the robot, and the end-effector. Thus, the truss element decomposition is sequentially directed toward separate targets. Each of the device commands, such as the end-effector install command (INSTALL) decomposes into a sequence of individual actuator commands (CLOSE, LOCK, EXTEND,...). If a failure is detected, automated error recovery is activated. If the automated error recovery is not successful, an error recovery menu is displayed to the operator. On the menu is a list of methods to overcome the failure. The order of the list and the definition of the methods are based on experience. The system will not proceed until the problem is resolved. If it cannot be corrected, information is passed back up through the system hierarchy, causing all commanded actions to roll back to the state that existed prior to initiating the command. More information regarding fault corrections and overrides, especially with regard to the end-effector, will be discussed in a subsequent section. Further details on the facility components and the software system can be found in reference 3.

DESCRIPTION OF ASSEMBLY HARDWARE

The second generation end-effector discussed throughout this paper is a follow-on to the first generation end-effector used to assemble the planar structure shown in figure 1. This end-effector is shown in the center of figure 5 surrounded by enlargements of the end-effector components. The second generation end-effector is much smaller than the first generation model, with a length of about 0.52 meters (20.5 inches). It is designed to insert one end of a strut at a time. The receptacle fingers, nut driver, insertion platform, and strut holder are adopted directly from the first generation end-effector. The second strut holder (figure 5b) and alignment fingers (figure 5c) were added to increase the moment capacity and capture zone of the end-effector. The alignment platform permits the alignment fingers to be extended away from the strut holders when capturing cantilevered struts.

The end-effector contains four grippers, two platforms, a nut driver, and seventeen sensors (not shown in the figure). The insertion platform, shown in figure 5e, is used to insert the strut connector into the node receptacle. The alignment platform is used to extend the alignment fingers to prevent interference with other end-effector components when capturing a cantilevered strut, as mentioned above. Referring to figures 5e and 5f both platforms will extend and retract. These operations are accomplished by air cylinders with full extension and retraction verified by linear potentiometers. Figures 5a and 5c depict the operation of the receptacle and alignment fingers. The receptacle fingers align the end-effector relative to the structure by closing on the vee grove of the node receptacle, the alignment fingers are used to capture a cantilevered strut by closing on the strut tube. Both the receptacle fingers and the alignment fingers are actuated by sliders driven by air cylinders. Position of the cylinder piston is monitored via the manufacturer's reed switches mounted on cylinder exterior. The reed switches are activated by a permanent magnet internal to the pneumatic cylinders. The receptacle and alignment fingers also contain infrared cross-fire sensors to verify that a strut is in the capture zone. The cross-fire sensors operate using a pair of units that project light from an emitter unit to a receiver unit. A receptacle or strut is detected when it blocks the light. Figure 5b depicts operation of the strut holders. The strut holders are driven by DC motors through a screw jack mechanism. The strut holders align the end-effector relative to the strut by closing on the strut's alignment adapter. Motor power is applied to the strut latch motors until a sensor is activated or the time limit is exceeded. Three inductive sensors are used to monitor each strut holder's state: one sensor is used to indicate that the holder is FORCED CLOSED, and two to sense the state of the motor-driven latching mechanism, either DRIVEN OPEN or DRIVEN CLOSED. The strut holders are spring loaded to the open position. FORCED CLOSED occurs when the robot is commanded to move until the strut holders are forced closed by contact with the alignment adapter located on the strut, then the strut holders are DRIVEN CLOSED securing the strut to the end-effector.

The nut driver, figure 5d, is also driven by a DC motor. The nut driver is used to actuate the joint connector. The nut driver is driven using a closed loop control system with current and position feedback. Current feedback is derived from the voltage drop across a resistor in series with the motor winding. Position feedback is achieved by a low resolution encoder on the output shaft providing eight pulses per revolution. Since the flats of the socket and the nut are randomly oriented during acquisition of the strut, the socket will not necessarily slip onto the nut, but instead may push the strut away from the holders, preventing them from grasping the strut. The socket is therefore spring loaded toward the strut to permit grasping of the strut even though the nut driver may not be seated. Socket seating is monitored by an inductive sensor.

STRUT INSTALLATION

The second generation end-effector must be able to install struts into nodes supported by one or more previously installed struts. As an example of the operations required for strut installation, a possible double cantilever installation scenario will be developed. In this case the end-effector has installed one end of a strut and now must connect the other cantilevered end to an existing node at the end of a second

cantilevered strut. A cross-fire sensor verifies that a strut is within the capture zone of the alignment fingers, figure 6a, the alignment fingers are closed on the cantilevered strut's tube restricting the translation of the strut in two directions. With the strut captured and the proper alignment achieved, figure 6b, the insertion platform is extended and strut holders latched securing the strut to the end-effector, figure 6c. The left strut holder closes around the hexagonal alignment adapter that is bonded to the strut tube. These adapters have a central vee groove that mates with a protrusion in the left strut holder to provide passive axial alignment, while the hexagonal cross section of the adapter keeps the strut circumferentially aligned. The right strut holder closes on the strut tube to increase the moment capacity of the end-effector's grasp. At this time the alignment fingers are opened, figure 6d, followed by the retraction of both platforms. With the cantilevered strut secure, the next step is the capture of the node at the end of the second cantilever strut. After verifying the receptacle is within the capture zone of the receptacle fingers using the cross-fire sensor, the fingers are closed on the vee groove machined into the connector receptacle, passively correcting small misalignments. The end-effector fingers are designed to successfully capture the receptacle within a cylindrical envelope of 5 cm diameter by 1.5 cm long. With the receptacle captured, figure 6e, the insertion platform is extended inserting the joint connector into the receptacle. The nut driver socket is seated; if required, by incrementally turning the socket to align it with the nut, then several full turns of the nut securely lock the joint. While the joint is being locked motor current, encoder counts, and socket seating are monitored. Normal termination occurs when current draw exceeds a predefined threshold and a minimum number of encoder counts have been received, while the socket has remained seated. Reaching the current threshold indicates the locking mechanism has applied adequate closure torque to the joint connector, while the encoder count indicates that the joint connector is fully closed. This operation secures the strut in the structure. Next, the strut holders are unlatched releasing the strut, the insertion platform is retracted, and the receptacle fingers are opened, releasing the structure and completing the connection. All operations are verified using sensors as discussed earlier. All other installation cases are based on variations of these operations.

The double cantilever installation is a potentially difficult task and several operational scenarios are possible. For example, the strut with the node may be captured first. The purpose of the above illustration is to demonstrate how the operation of the end-effector components must be coordinated to perform an installation sequence and how sensors are required to verify the operation of the end-effector components. It is important to note that all mechanical operations must be totally reversible to permit the end-effector to recover from an error or to permit the truss to be disassembled for repair.

END-EFFECTOR CONTROL SOFTWARE

The driving force for development of an on-board embedded microprocessor has been to reduce the number of signal lines from the end-effector to accommodate a quick change mechanism. The end-effector is essentially a tool for the robot and, since the robot is expected to perform many operations, it occasionally requires different tools. This led to the purchase of a quick change mechanism to allow the robot to engage or disengage end-effectors. But, this piece of hardware places a constraint on the number of signal lines between the end-effector and the robot. Thus, a method for integrating the sensors with the end-effector operations was required to accommodate the increasing number of sensors on the end-effector. This need lead to the development of the microprocessor based control scheme discussed in this paper.

An assessment of the requirements for the microprocessor system was conducted and the following microprocessor capabilities were identified: (1) a minimum number of signal lines passing through the quick change mechanism, (2) small physical dimensions, (3) standard communication support, (4) six input lines for analog to digital conversion, (5) two output lines for digital to analog conversion, (6) twenty-four general purpose I/O pins, and (7) three lines providing external interrupt capability. In addition, high level language and development tools were required to support the implementation of the following software capabilities: (1) stand alone checkout, (2) on line checkout, (3) communication interrupts, (4) operator override, and (5) eventual porting to different computer platforms.

These requirements led to the selection of a commercially available single board computer built around an eight bit microprocessor operating at a clock frequency of 12MHz. The board supports up to eight general purpose eight bit ports including two serial communication lines, external interrupts, data bus, and address bus. The board also supports eight analog to digital lines with programmable reference voltages and up to 64 kbytes of RAM and 64 kbytes of ROM.

To minimize the number of signal lines leaving the end-effector and provide necessary signal preparation a custom electronics board was fabricated to interface between the end-effector actuators/sensors and the microprocessor board. This board provides: (1) electronics for digital to analog conversion for proportional nut driver motor control, (2) electronics for analog to digital conversion to monitor nut driver motor current, (3) the drive electronics for the strut latch motors and solenoid valves, (4) electronics to convert plus and minus 24 volt power to the additional voltage levels required for the microprocessor and the drive electronics circuits, and (5) isolation of all sensor and actuator signals from the microprocessor ports by Schmitt triggers or voltage followers. By locating the support electronics on the end-effector, the number of signal lines required to operate the end-effector has been reduced from 60 to 5, providing enough free lines in the quick change mechanism for two color camera cables used by the operator to monitor end-effector operations.

The software was implemented in C using a PC based development system. The system was selected because it was ANSI C compatible, included a source level debugger, and included language extensions providing access to all processor dependent features. C was selected because of its support for bit level operations and to support transferring developed code to other computers.

The relationship of the end-effector to the overall assembly configuration is shown in Figure 7. Note that the second generation end-effector will communicate with the executive computer directly instead of through the robot as was depicted in figure 3. The end-effector control software is responsible for three levels of operation that were depicted earlier (figure 4); device, component, and verification. The software implementation focused on support for event driven operations. Event driven refers to program flow dictated by responses to events rather than moving serially through a state system. Event driven applications have many advantages. In general they provide improved response time, support asynchronous events, and provide interrupt capability. The ability to support asynchronous events and provide interrupt capability were essential for this application.

The relationship between end-effector commands and software states is shown in figure 8. Figure 8a is the state transition diagram for the end-effector software, while figure 8b is the corresponding transition matrix. In figure 8a arrows represent possible transitions, dashed arrows represent interrupts, and bold arrows represent normal operation.

Commands associated with a given transition overlay the arrow representing the transition. The shaded box on the left encloses the states making up the control loop. All commands that affect end-effector actuators occur within this box. The shaded box on the right side of the figure encloses the three states within the command parser. The software transitions from the box on the left to the one on the right via an interrupt activated by the reception of a command. When power is applied to the microprocessor's board, or the board's reset button is depressed, the software initializes (figure 8a) the actuators, based on sensor information, and enters the control loop state WAIT FOR COMMAND. Under normal_operation, the software is in the control loop in state WAIT FOR COMMAND when an executable command is received across the serial line activating the serial interrupt handler (command parser). The command parser reads the command, validates the command based on the current state, and initiates the transition from WAIT FOR COMMAND to EXECUTE COMMAND. After execution of the command has completed, the software waits for another command. The software design was governed by the possibility of being interrupted at anytime. For this reason and because the command parser initiates the transition from state WAIT FOR COMMAND to state EXECUTE COMMAND, the command parser and the commands it accepts will be discussed first.

The command interface to the microprocessor was designed to provide a human readable ASCII format. This greatly simplifies diagnostic and monitoring operations by creating data that can be immediately read and interpreted by the operator. This interface has the added advantage of supporting dumb terminals for off-line checkout and verification. For this verification to have relevance, it should mimic the conditions of the actual system. Thus, the commands issued from the terminal must be the same as those issued to the end-effector at run time. By carefully defining the syntax of the end-effector commands, the overhead is minimized while maintaining a human readable format. Under normal operation the maximum command contains seven ASCII characters.

The most critical requirement for asynchronous event support was to allow PAUSE and REVERSE commands at any point. PAUSE implies suspending the current operation indefinitely. REVERSE implies removing the progress of the current operation until the state of the system is the same as it was before the command was received. As shown by the dashed lines in figure 8a, command reception activates the command parser. While the software is in the state EXECUTE COMMAND, reception of a PAUSE command will cause the software to enter the PAUSED state suspending execution indefinitely. Either a CONTINUE or a REVERSE command is required to resume execution.

The correspondence between the commands and software state transitions is shown in figure 8b. The states that can be interrupted are located on the left of the figure, the acceptable commands in the center, and the resulting state on the right. Commands are interpreted based on the current software state, thus there are three cases of command evaluation within the command parser, one for each software state that may be interrupted by command reception. Interruption occurs when input is detected on the serial line, because the command parser (serial interrupt handler) is activated to process it, thus, allowing commands to be received at anytime. Note that the command parser contains all decision points related to software flow.

Supervisory commands can be entered from any state and do not cause a state transition. Supervisory commands provide a method for direct operator interaction with the end-effector. For example, a single step mode may be turned on. This causes the end-effector control software to execute component commands individually. After completion of each component command, the operator is queried to continue. Supervisory commands also support diagnostic operations. They give the operator a way of requesting end-effector status information to monitor individual sensors. Consider the platforms discussed earlier; if they indicate intermittent failures when a platform actually worked, then the platform potentiometer may be monitored to determine if the thresholds representing fully extended or retracted need to be adjusted. After completion of the supervisory action, control returns to the state that existed prior to the reception of the supervisory command.

While in the software state WAIT FOR COMMAND, executable commands are accepted. Executable commands initiate end-effector action, thus causing transition to the state EXECUTE COMMAND. For example, LOCK NUT is an executable command that causes the nut motor to turn. CONTINUE and REVERSE are also accepted in the state WAIT FOR COMMAND. Reception of these commands implies the previous command encountered an error, because an unresolved error terminates command execution. CONTINUE implies the error has been overcome and execution is to resume, REVERSE implies the error was not overcome and the system must be rolled back to the state before the command was received.

From the state EXECUTE COMMAND, reception of REVERSE implies the operator has foreseen a potential problem and wishes to retract the last EXECUTABLE command. If PAUSE is issued by the operator, while in the end-effector software state EXECUTE COMMAND, the end-effector will suspend its current operation prior to entering the PAUSED state. To suspend the current end-effector operation, the control software uses a queue of pause functions. These are functions queued by the component or composite functions to be activated anytime a PAUSE is received. For example, if the PAUSE command is received while locking the nut, the nut driver motor must be stopped. This is accomplished using a function queued by the nut component that stops the motor and saves the encoder counts. After activating each function in the pause queue, the control software enters the PAUSED state. The end-effector may remain paused indefinitely. As shown at the bottom of figure 8b, execution of the interrupted end-effector operation, i.e. return to the state EXECUTE COMMAND from the state PAUSED, will occur only after reception of a CONTINUE or REVERSE command. Queues similar to the pause queue are maintained for CONTINUE and REVERSE. If CONTINUE is received, continuing from the PAUSED state entered above, a function in the CONTINUE queue is activated to restore the encoder counts and restart the nut motor, thus returning to the state before the PAUSE command was received. If an invalid command is entered at any time an error is returned.

The control loop, shown on the left of figure 8a is shown in detail in figure 9, with the corresponding portions of the software flow diagram enclosed in circles. The block ACTIVATE FUNCTION is further detailed in the lower right of the figure. After power is applied to the microprocessor board or a reset occurs, the end-effector software outputs a message, signaling successful initialization, and waits for a command to execute. At this time the software is in an infinite loop, constantly checking a flag that, when TRUE, indicates a command is available to execute. This flag can only be set to TRUE through an interrupt handler. As mentioned above, the command parser is an interrupt handler. By setting the flag to TRUE after a valid command has been received, the command parser initiates the transition from the state WAIT FOR COMMAND to the state EXECUTE COMMAND.

Commands are of two types, component and composite. Both are activated and processed in the same fashion within the control loop as shown in figure 9. Component commands interact with the hardware directly and hence, are specific to a given end-effector. All component commands contain a timed loop during which a specific sensor is polled to verify success of the operation. Three basic capabilities are provided for each

end-effector component by component commands. These capabilities are: (1) specifying the desired configuration of the component, for example EXTEND INSERTION; (2) exercising the component, for example CYCLE INSERTION, which attempts to extend and then retract the insertion platform; and (3) analyzing the component function, for example SURVEY INSERTION, which cycles (for a predefined number of times) the insertion platform and displays a series of formatted strings containing the commanded state, the thresholds representing fully extended and retracted, and the current value of the linear potentiometer.

An example component function block diagram is shown in figure 10. This component is responsible for receptacle finger operations. For each command available for the component, a section of code is required to define the command operation. In this example the executive command CLOSE RECEPTACLE was received. Upon entering the section of code for CLOSE, a sensor check is made for receptacle presence via infrared cross-fire sensors. An error is generated if the receptacle is not present. If a receptacle is present then the actuator bit corresponding to the receptacle fingers' actuator is set. The software then enters a timed loop during which the reed switch, which indicates closed, is constantly polled. If the receptacle fingers close before the allotted time has expired the operation is successful, and a message indicating success is returned by the control loop. Time limits used in the component commands were established empirically based on experience with the first generation end-effector. If the receptacle does not close within the allotted time an error code is returned.

In the command hierarchy, composite commands are above component commands. Composite commands are a set of end-effector specific component commands executed in a predefined order. Composite commands allow for a greater level of abstraction, thus, they provide a method for increasing the embedded intelligence at the end-effector level. Composite commands were established to provide a device independent interface to the end-effector operations. Use of composite commands has significantly reduced the code requirements in the executive program by standardizing the interface to all end-effectors. Also, by supporting the composite commands at the end-effector level, most of the communication overhead has been eliminated. The block diagram for an example composite command is shown in figure 11. This is the sequence for a subset of the double cantilever INSTALL function. In the explanation that follows, it is assumed that the end-effector already has a strut latched by the strut holders and the insertion platform is retracted. Because any command performed by the end-effector must be reversible, each composite function has a corresponding reverse function, in this case INSTALL_REVERSE, to return the end-effector to the state prior to reception of the composite command. Both functions are depicted in the figure; INSTALL on the left, INSTALL_REVERSE on the right. An INSTALL operation with no errors will enter at the top, the continue check will be FALSE, and the operations from SET_UP to COMPLETE will occur sequentially. In SET_UP the receptacle is verified to be present, then the receptacle fingers are closed in CLOSING_RECEPTACLE. Next a message is sent to the executive program requesting a force/torque balance. Force/torque balancing will be used to adjust the robot arm to correct for misalignments between the end-effector and receptacle slot that prevent full extension of the insertion platform. The robot is adjusted until all reaction forces between the robot and structure have been reduced to near zero. At this point the end-effector will be in the configuration shown in figure 2e. As discussed previously, the insertion platform is then extended and the nut locked to secure the strut. Finally the strut is unlatched, the insertion platform retracted, and the receptacle fingers opened, releasing the structure and completing the installation sequence. A sensor is polled after each operation to verify successful execution. Because no resources have been allocated, no operations occur in COMPLETE. The software then returns to the control loop where a message indicating success is returned to the executive program.

Software supporting error handling has dominated the implementation efforts for the composite commands. In general, approximately 70% of the software is required to support error conditions. If an error occurs in one of the component operations; the function is terminated, returning to the main control loop where a descriptive error message is returned to the executive program. After receiving an error message the executive program presents an error menu to the operator through which the operator can: (1) fix the error and continue, (2) reverse the current command, or (3) override the error and continue. It is important to note that, after an error is encountered, the function is terminated and the end-effector software is in the control loop state WAIT FOR COMMAND. This provides two advantages. First, since there is only one command parser, each function does not have to support communications during error handling. Second, there is no restriction on the commands that can be issued when trying to resolve an error. This is critical, because it is often necessary for the operator to activate other end-effector components or command other devices when resolving an error. For example, when closing the receptacle, the end-effector's location may need to be modified by commanding the robot to move. Thus, the receptacle is opened, using a component command, the robot moved, and the installation sequence resumed by a CONTINUE command.

If the operator is able to eliminate the error by adjusting some assembly component, then a CONTINUE command is issued to the end-effector. The control software sets the continue bit too TRUE, causing the CONTINUE? check at the beginning of INSTALL to be TRUE, thus returning to the state within the INSTALL function where the error occurred and continuing execution from that state.

If the operator is unable to overcome the error, the REVERSE command is sent to the end-effector. The end-effector software sets the flag indicating CONTINUE, but also sets a flag causing the components to abort. When the last state is entered, it is immediately aborted, returning a unique error condition to the INSTALL function. The INSTALL function clears the error before calling the INSTALL_REVERSE function. INSTALL_REVERSE jumps to the current state and begins reversing the operations completed by the INSTALL function. If another REVERSE command is received, then the same error is cleared by the INSTALL_REVERSE function and control is returned to the INSTALL function. This cycle can repeat indefinitely. Jumps to the current state are depicted by large arrows under the line TO CURRENT STATE at the bottom center of figure 11. Note that some of the states in INSTALL_REVERSE have no actions associated with them, for example BALANCING_FTS, which is only required after the end-effector has grasped the structure. These states are simply place holders so that when reversing there is an appropriate state to jump to. An example of a reverse sequence follows. While in the INSTALL state CLOSING_RECEPTACLE, the receptacle fingers fail to close, the timing loop initiates an error causing the INSTALL function to terminate and return to the control loop, where an error message is sent to the executive program. The end-effector software then enters the state WAIT FOR COMMAND. Because the operator is unable to overcome the error, the REVERSE command is sent to the end-effector. The end-effector control software interprets the REVERSE command, sets the abort flag to TRUE, sets the CONTINUE flag to TRUE, and activates the INSTALL function. Upon entering the INSTALL function the CONTINUE check is TRUE, causing execution to return to the state

CLOSING_RECEPTACLE. When the component command responsible for receptacle operations is executed, it immediately aborts, because the abort flag is TRUE, returning a unique error code to the INSTALL function. The INSTALL function clears the error before calling the INSTALL_REVERSE function, thus reversing the sequence. While in the INSTALL_REVERSE function, reception of another REVERSE command will result in the same error code being cleared by the INSTALL_REVERSE function causing execution to return to the INSTALL function. Cycling between INSTALL and INSTALL_REVERSE can repeat indefinitely. The REVERSE command may also be received while the software is in state EXECUTE COMMAND. Reception of the REVERSE command from this state results in the abort flag being set to TRUE causing the current component operation to abort. The unique error condition is cleared and the REVERSE function is executed as described above.

Finally, it is possible to override an error and continue the execution of the command. This capability is provided by a function called UPDATE. UPDATE can be thought of as a background process, because it is never called directly. UPDATE operates as an interrupt handler cycling at 15 Hertz using an autoreload timer available on the microprocessor. UPDATE provides four basic functions: (1) a method of overriding sensor information, (2) support for asynchronous notification of hardware changes, (3) a method for monitoring sensor health, and (4) isolation of the software from the computer hardware to ease the transfer of the software to other computers. In addition UPDATE is the function responsible for interpreting the sensor data and updating the database, as well as interpreting the command settings and updating the database and end-effector actuator settings.

The capability to override sensor information, requires a database independent of the end-effector hardware. When power is applied to the microprocessor, or the microprocessor boards reset button is depressed, the software initializes the database based on the current end-effector configuration. Thus, the information in the database represents the current state of the end-effector hardware and software control flags. However, because this database is independent of the end-effector hardware, the information from the senors may be ignored by simply not updating the database. The M_VAX_CONTROL mask is used to prevent updates of the database and actuator settings. Bits set in this mask are not modified, thus indicating the operator has taken responsibility for the setting of one or more sensors. NULLIFY is the supervisory command used to initiate sensor override. The end-effector control program uses the last error status to set the database variable so that no error condition will be generated, and sets bits in M_VAX_CONTROL to prevent UPDATE from changing the value in the database. For example, if the receptacle fingers did not close, then this error will be overridden as follows. First the sensor bits in the M_VAX_CONTROL mask corresponding to the sensor which detect finger closure and the sensor that detects fingers opening are both set to prevent UPDATE from modifying these sensor bits in the database. Then the bits in the database are set directly to indicate the receptacle fingers are closed. The open sensor is set to FALSE and the close sensor is set to TRUE. Note that the order is critical to prevent conflicts which occur if both bits are TRUE simultaneously. At this point the database is configured so that the receptacle fingers appear closed to the software. The next command to close the receptacle fingers will succeed, thus successfully overriding the FAILED TO CLOSE error.

Support for asynchronous notification of hardware changes is provided by a third mask, the M_WATCH mask. If a watched value changes, a software interrupt is activated, thus notifying the control software of the change. For example, for correct operation of the end-effector, the supply voltages must remain nominally constant. Monitoring supply voltage could be implemented using the M_WATCH mask. If the voltage changes, the watch function is called to compare the voltage level to predefined thresholds. If the voltage is out of range the end-effector could be switched to battery backup and the executive program notified so that corrective actions could be initiated. This capability has not been implemented to date.

UPDATE provides the capability for monitoring sensor health. Currently only conflicts are reported. Conflicts occur between sensors that are configured so that both cannot be true simultaneously. For example, the receptacle fingers are designed so that they cannot simultaneously activate the two sensors that indicate CLOSE and OPEN. If this occurs, the hardware has failed, either mechanically or electrically, and this failure is reported to the host as a sensor conflict. This capability can be expanded to provide on-line sensor reconfiguration, which would require redundant sensing capability that is not available at this time. Note that; in general, two sensors that disagree provide only the information that the signal from one sensor is invalid because of some form of failure. There is no information about the state of the operation being sensed. If the operation of the system has been monitored, and the system has performed as expected, then the sensor that disagrees with the command signal is probably invalid, because it is unlikely that both a sensor and an end-effector operation will fail simultaneously. If three sensors are available; then the two that agree are considered valid, while the remaining sensor is considered invalid and flagged for maintenance.

UPDATE isolates the software executing on the microprocessor from the microprocessor specific features. It is one of two microprocessor specific functions. The other is used to initialize the microprocessor board after turning on power or after a reset. Thus only two functions need to be modified if it is desired to move to a different processor.

TESTS AND CURRENT OPERATIONAL STATUS.

Currently the end-effector hardware and software are undergoing integration tests in an off-line checkout mode. All support software, including the composite sequences, have been implemented. The end-effector will then undergo extensive testing on a six-axis servo table to refine the operational scenarios required to assemble the tetrahedral structure, including the double cantilevered capture and insertion operations described earlier. During this process, common errors will be logged along with their corrective actions. This will be used to add self-correcting logic to each of the composite commands. The outcome of these tests should finalize the component commands and the sequences used in the composite commands.

After development testing is complete on the six-axis servo table, the end-effector will be used to assemble the two ring structure shown previously in figure 1. Following successful assembly, operations will likely progress to the assembly of structures with curved surfaces, which are assembled using straight struts with different lengths.

Work has already begun applying these routines and techniques to other end-effectors used in the assembly facility. In support of these activities, generic code models are being developed for actuator/sensor groups. Not only will these techniques be applied to the end-effectors of the facility, but to other devices in the facility, such as the motion bases and pan tilt drive units to position video cameras. These distributed dedicated microprocessors provide an initial step toward the realtime distributed architecture needed for efficient control of automated assembly operations.

SUMMARY

NASA Langley Research Center has initiated a research program to develop methodologies for automated in-space assembly of large truss structures. A laboratory test facility, called the Automated Structures Assembly Laboratory, has been recently developed in which a regular tetrahedral truss structure, is comprised of 102 struts, has been assembled using a specialized end-effector mounted on an industrial robot arm. To expand applications that can be addressed by the facility, an end-effector which installs one end of a strut at a time and uses a dedicated on-board microprocessor for sensor interrogation and actuator control, was developed and fabricated. The new end-effector will permit assembly of a variety of structures, installation of standardized payloads, and will provide the first opportunity for concurrent operations within the test facility.

The new end-effector has been fabricated, and most of the control software written. The end-effector adopts many features from the first generation end-effector. The new end-effector contains four grippers, two platforms, and a nut driver. The end-effector is directly controlled by an embedded microprocessor. The software supports asynchronous interrupts to allow the operator to suspend or reverse end-effector operations at any time.

The end-effector control software development was dominated by the need to support error conditions and asynchronous events. Thus, efficient means of interrupting software execution and servicing asynchronous events are essential, both when interacting with human operators and when interacting with end-effector hardware. The end-effector software development effort has isolated and standardized the interface to the end-effectors used in the facility, resulting in a significant code reduction and generalization of the executive program.

The embedded microprocessor and support electronics board has reduced the number of signal lines required to operate the end-effector from 60 to 5, allowing the exchange of end-effectors via a quick change mechanism. Also, addition of an embedded microprocessor to the end-effector currently used in the facility is expected to significantly reduce assembly time by supporting concurrent operations and eliminating most of the time required for end-effector communications.

The second generation end-effector will greatly increase the flexibility of the Automated Structures Assembly Laboratory. This end-effector development effort represents a first step toward the realtime distributed computational architecture needed to provide efficient control for automated assembly systems.

REFERENCES

1) Wu, K. Chauncey; Adams, Richard R.; Rhodes, Marvin D.: "Analytical and Photogrammetric Characterization of a Planar Tetrahedral Truss", NASA Technical Memorandum 4231, December 1990.

2) Rhodes, Marvin D.; Will, Ralph W.; Wise, Marion A.: "A Telerobotic System for Automated Assembly of Large Space Structures", NASA Technical Memorandum, March 1989.

3) Will, Ralph W.; Rhodes, Marvin D.: "An Automated Assembly System for Large Space Structures", SPIE Symposium on Advances in Intelligent Systems, Cooperative Intelligent Robotics in Space, November 1990.

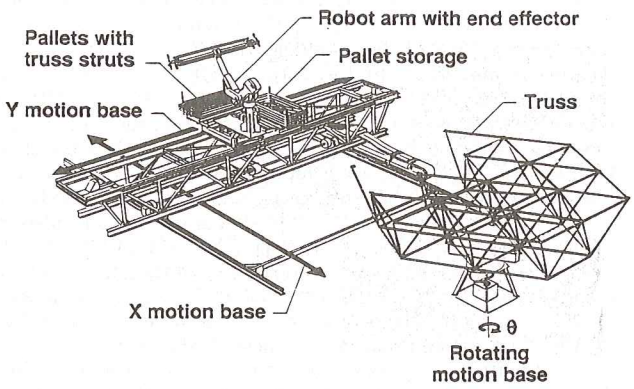


Figure 1a
Schematic of Automated Assembly Facility.

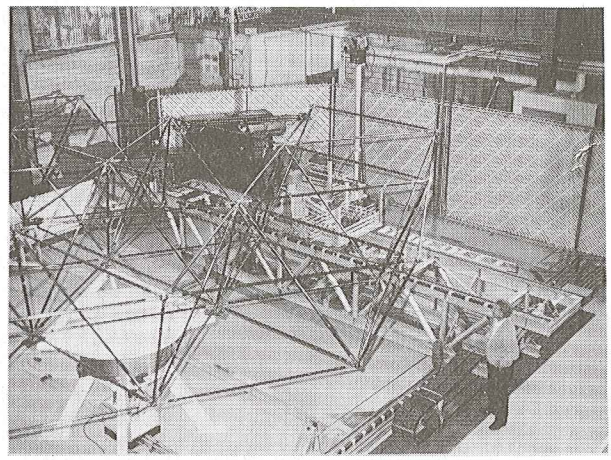


Figure 1b
Photograph of Assembly Facility.

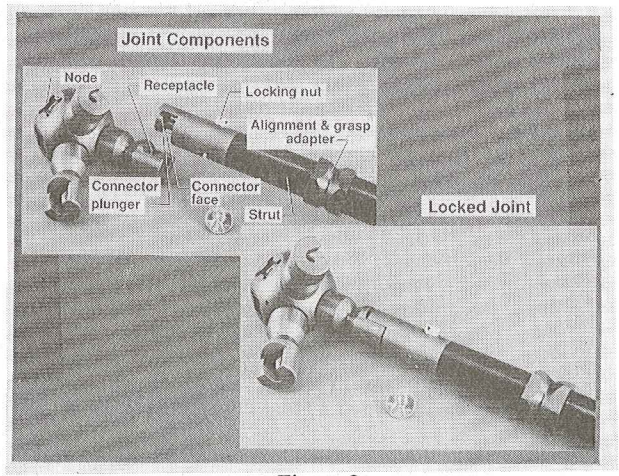


Figure 2
Photograph of Truss Node and Joint Connection Hardware.

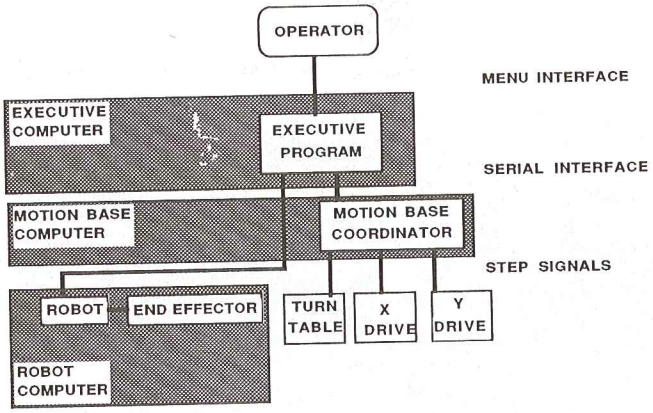


Figure 3
Current Assembly Facility Computer and Software Architecture.

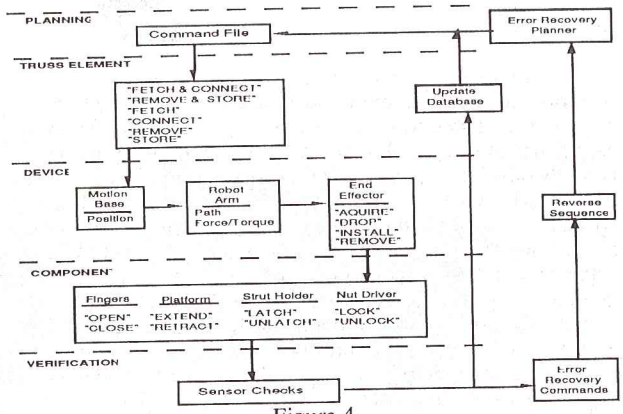


Figure 4
System Levels and Command Hierarchy

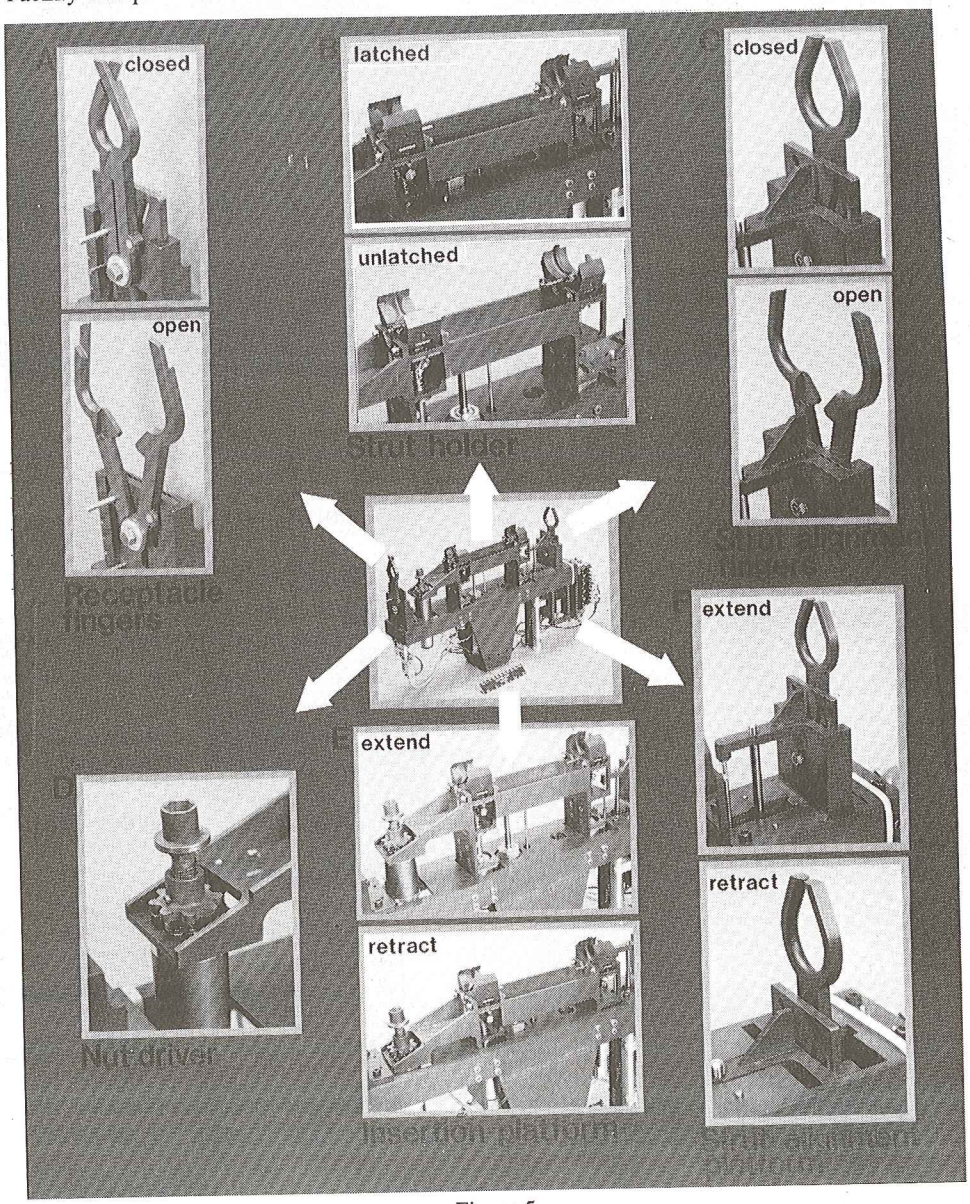


Figure 5
Second Generation End Effector.

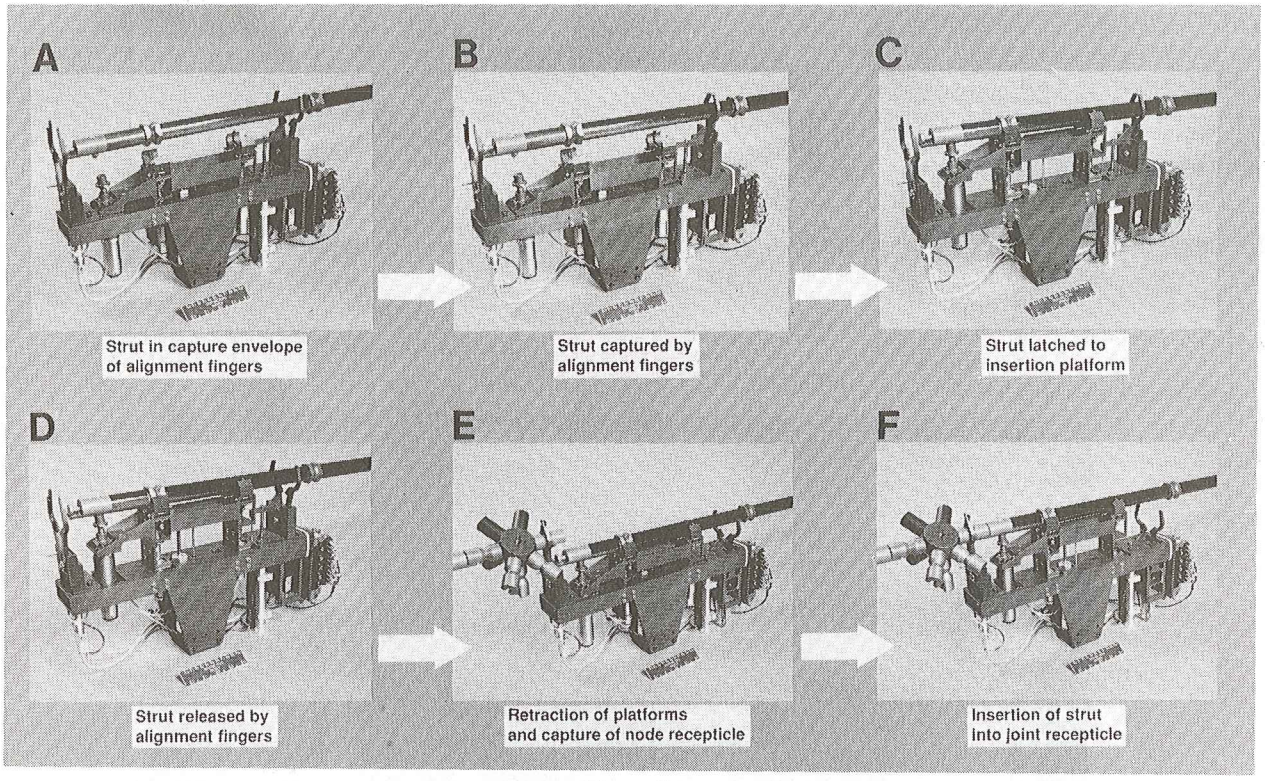


Figure 6
Capture and Installation of a Strut.

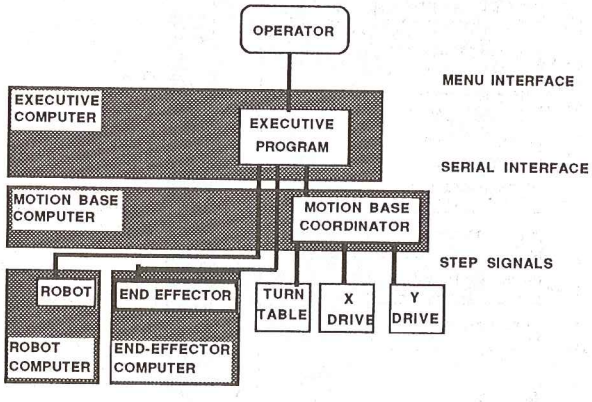


Figure 7
Current Assembly Facility Computer and Software Architecture.

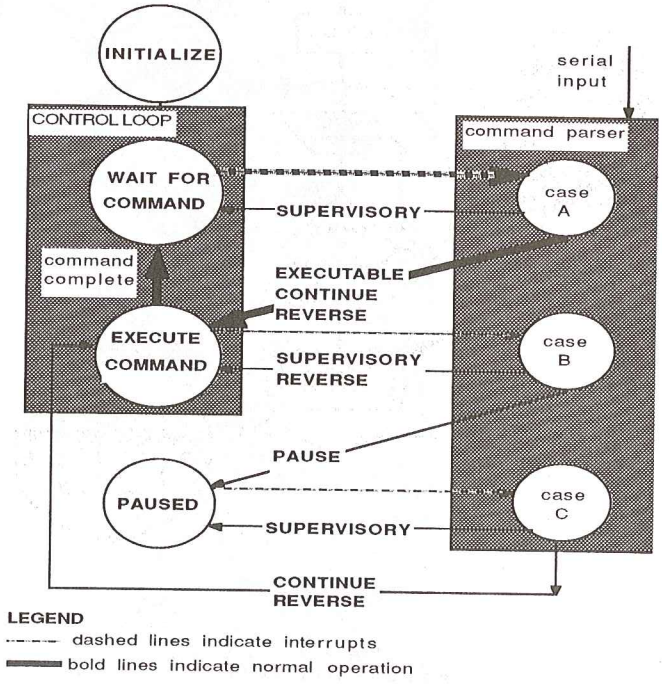


Figure 8a
Software State Transition Diagram.

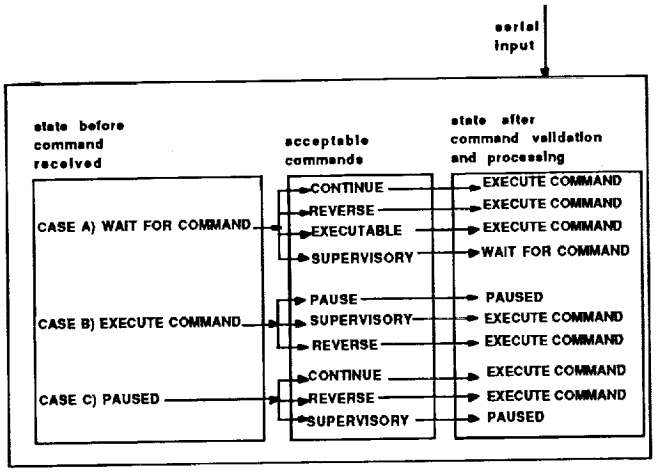


Figure 8b
Software State Transition Matrix.

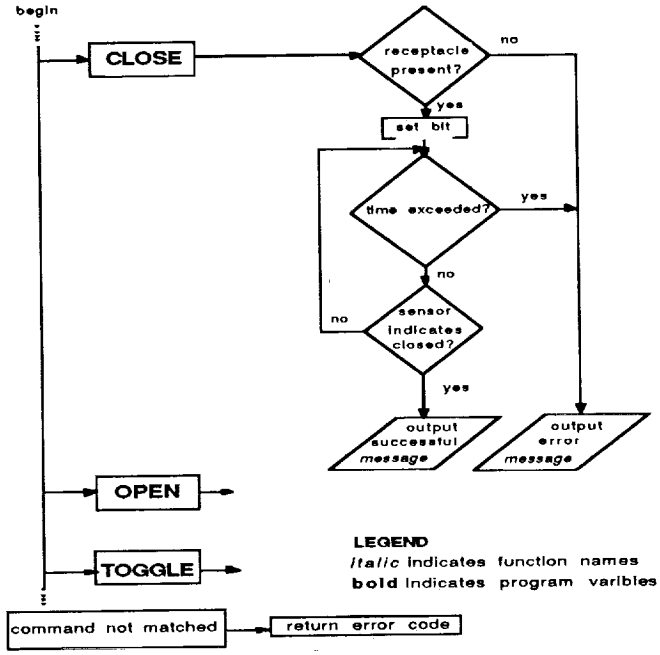


Figure 10
Block Diagram of Function to Execute Typical Component Command
(Receptacle)

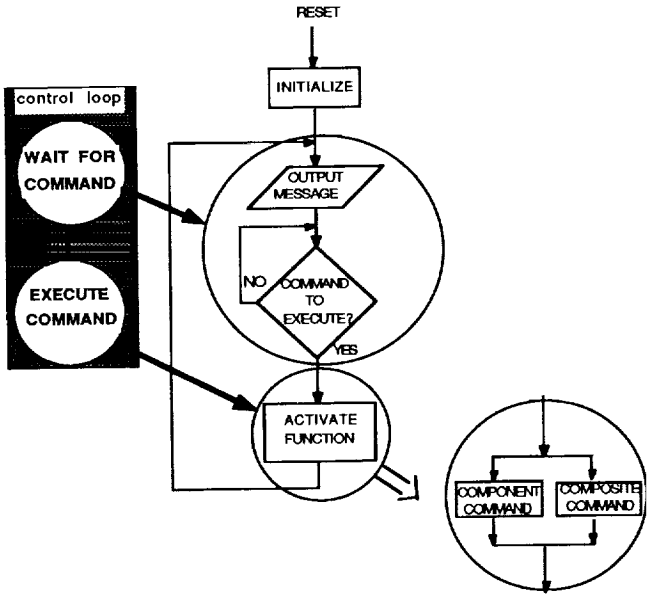


Figure 9
Software Control Loop.

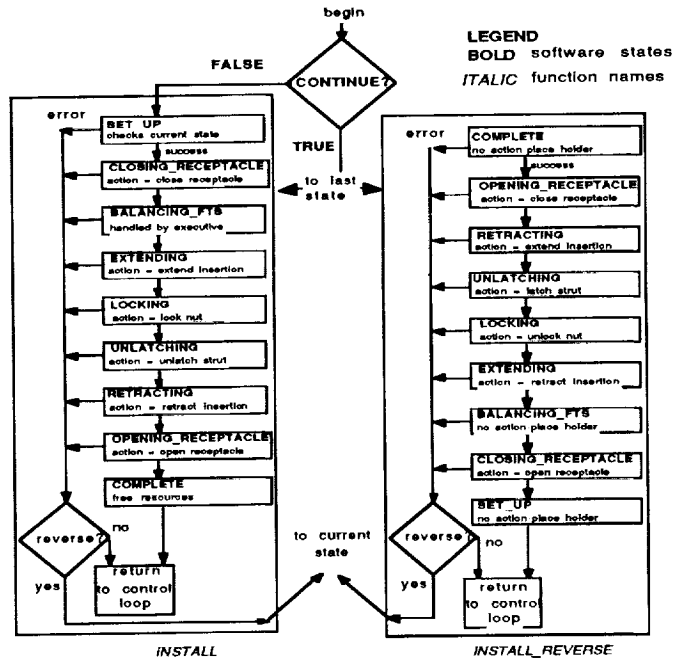


Figure 11
Block Diagram of Function to Execute Typical Composite Command
(Portion of Install)

AUTHOR INDEX

★U.S GOVERNMENT PRINTING OFFICE: 1992 - 627 - 064/46012

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1992	Conference Publication

4. TITLE AND SUBTITLE
Fifth Annual Workshop on Space Operations Applications and Research (SOAR '91)

5. FUNDING NUMBERS

6. AUTHOR(S)
Kumar Krishen, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lyndon B. Johnson Space Center
Houston, Texas 77058

8. PERFORMING ORGANIZATION REPORT NUMBER
S-650

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546
U.S. Air Force, Washington, D.C. 23304
University of Houston - Clear Lake, Houston, Texas 77058

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CP-3127, Vol. I

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category: 59

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document contains papers presented at the Space Operations, Applications and Research Symposium, hosted by NASA Johnson Space Center (JSC) and held at JSC in Houston, Texas, on July 9-11, 1991. More than 110 papers were presented at this Symposium, sponsored by the U.S. Air Force Phillips Laboratory, the University of Houston-Clear Lake, and NASA JSC. The technical areas covered were Intelligent Systems, Automation and Robotics, Human Factors and Life Sciences, and Environmental Interactions. The U.S. Air Force and NASA programmatic overviews and panel discussions were also held in each technical area. These proceedings, along with the comments and suggestions made by the panelists and keynote speakers, will be used in assessing the progress made in joint USAF/NASA projects and activities. Furthermore, future collaborative/joint programs will also be identified. The SOAR '91 Symposium is the responsibility of the Space Operations Technology Subcommittee (SOTS) of the USAF/NASA Space Technology Interdependency Group (STIG). The Symposium proceedings includes papers covering various disciplines presented by experts from NASA, the Air Force, universities, and industry.

14. SUBJECT TERMS
Space, operations, automation, robotics, life sciences, medical protocols, knowledge acquisition, expert systems, radiation, biomedical, debris, space shuttle, Space Station Freedom, lunar outposts, Mars missions.

15. NUMBER OF PAGES
464

16. PRICE CODE
A20

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration
Code NTT

Washington, D.C.
20546-0001