NASA Conference Publication 3103, Vol. I

Fourth Annual Workshop on Space Operations Applications and Research (SOAR '90)

Robert T. Savely, *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 New Mexico,
Albuquerque, New Mexico, and held in
Albuquerque, New Mexico
June 26–28, 1990

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division
1991

CONTENTS

PREFACE

The papers presented at the Space Operations, Applications, and Research (SOAR) Symposium, hosted by the Air Force Space Technology Center and held at Albuquerque, New Mexico, on June 26-28, 1990, are documented in this proceeding. Over 150 technical papers were presented at the Symposium which was jointly sponsored by the United States Air Force (USAF) and the NASA/Johnson Space Center. The technical areas included were: Automation and Robotics, Environmental Interactions, Human Factors, Intelligent Systems, and Life Sciences. NASA and USAF programmatic overviews and panel sessions were also held in each technical area. The Symposium proceeding includes papers presented by experts from NASA, the USAF, universities, and industries in various disciplines. These proceedings, along with the comments by technical area coordinators and session chairmen, will be used by the Space Operations Technology Subcommittee (SOTS) of the Air Force Systems Command and NASA Space Technology Interdependency Group (STIG) to assess the status of the technology, as well as the joint projects/activities in various technical areas.

MESSAGE FROM THE GENERAL CHAIR AND ASSISTANT GENERAL CHAIR

The SOAR 90 workshop will bring together Air Force and NASA project/program managers and members of the technical community for an information exchange on space operations. The visibility the SOAR 90 workshop will provide attendees into technology applications will give us a singular opportunity to establish additional cooperative technology development and transition programs.

As you can see, SOAR now represents Space Operations, Applications, and Research. This change reflects the nature of the conference which includes sessions on many areas in addition to automation and robotics. Each session will start with Air Force and NASA programmatic overviews of present efforts which will be followed by technical papers and conclude with panel discussions of problems/ solutions within the topic area.

With your participation, we look forward to an informative and productive workshop.

Col. Paul C. Anderson,
Air Force Space Technology Center

As you noticed, SOAR now stands for Space Operations, Applications and Research, and encompasses the broad scope and excitement felt by the space community as our Nation's revitalized space program gains strength and stability. The planned civil space scenario includes Space Station Freedom, a multitude of science and technology missions, the lunar outpost, and Mars missions/outposts.

The consequences of these space programs will be an infrastructure of space, lunar and Mars surface and ground operations. These operations need to be conducted effectively, efficiently, and with utmost safety. Advanced techniques/ technology are needed to enable future space operations to be carried out with these attributes.

SOAR 90 continues, with amplification, the theme of previous SOAR workshops in the discussion of future techniques/technology and approaches needed for space operations. Also, life sciences was added to the technical areas for the first time this year to reflect the broad scope of the needed research. The benefits to NASA and the Air Force in expanding communications and identifying cooperative programs have made SOAR an invaluable yearly event.

Dr. Kumar Krishen,
NASA Johnson Space Center

AUTOMATION AND ROBOTICS

MANIPULATION I

N91-20642

RESOLVED RATE AND TORQUE CONTROL SCHEMES FOR LARGE SCALE SPACE BASED KINEMATICALLY REDUNDANT MANIPULATORS

Robert W. Bailey
LinCom Corporation
1020 Bay Area Blvd, Suite 200
Houston, Tx 77058

Leslie J. Quiocho
Robotic Systems Evaluation Branch - ER4
Automation and Robotics Division
NASA/Lyndon B. Johnson Space Center
Houston, Tx 77058

ABSTRACT

Resolved rate control of kinematically redundant ground based manipulators is a challenging, but well understood problem. The structural, actuator, and control loop frequency characteristics of industrial grade robots generally allow operation with resolved rate control -- a rate command is achievable with good accuracy. However, space based manipulators are quite different, typically having less structural stiffness, more motor and joint friction, and lower control loop cycle frequencies. These undesirable characteristics present a considerable Point of Resolution (POR) control problem for space based, kinematically redundant manipulators for the following reason: a kinematically redundant manipulator requires an arbitrary constraint to solve for the joint rate commands. A space manipulator will assuredly not respond to joint rate commands as expected because of these undesirable characteristics. The question is, will low frequency rate feedback be adequate for POR control, and if not, what type of control scheme will be adequate? A space based manipulator simulation, including free end rigid body dynamics, motor dynamics, motor stiction/friction, gearbox backlash, joint stiction/friction, and Space Station RMS type configuration parameters, is utilized to evaluate the performance of a well documented resolved rate control law. Alternate schemes involving combined resolved rate and torque control are also evaluated.

INTRODUCTION

Space based manipulator design imposes a more stringent control problem than does ground based design. Space manipulators must be as light as possible to avoid excessive earth to orbit transportation costs. Space manipulators must also be able to survive the vacuum conditions on orbit. These design requirements produce manipulators with light-weight structures, large gear-ratios (relatively large backlash regions), high frequency low torque motors, excessive joint and motor friction, and low frequency control loop cycles (flight qualified computers which are slow compared to today's technology). To compound these physical contributions to the control problem, a kinematically redundant manipulator inherently allows an infinite number of arm configurations to achieve a given Point Of Resolution (POR) Cartesian position and orientation. This redundancy has its advantages and disadvantages. One such advantage is that infinite joint solutions allows a diverse range of control schemes. However, a disadvantage is that it can contribute to arm configuration drift through a given POR maneuver. In this paper, we will conceptually explain this problem and support the explanation with dynamic and kinematic simulation analysis of a Space Station RMS type manipulator. Variations of the classic motor rate feedback control system based on POR force and torque control will also be discussed as a possible solution to the problem.

PROBLEM CONCEPTUALIZATION

The majority of Space Station assembly analysis is currently being performed with kinematic manipulator simulations. Assuming that the actual space-based manipulator will respond exactly as the kinematically simulated manipulator, operational scenarios developed using the kinematic simulation will be adequate for space operations. This assumption could lead to dangerous consequences if kinematic control is not augmented with dynamic simulation. Several aspects of space manipulator operations will contribute to a drift (compared to kinematic simulation response) in the arm configuration through a given POR maneuver. Note that the problem of a drifting arm configuration from expected results is an entirely different problem from fundamental control of the manipulator POR.

Space based manipulator joint state responses have the following characteristics:

1) discontinuities - due to joint and motor stiction,
2) non-linearities - due to motor gearbox backlash (flexibility), and
3) variations - due to changing mass properties of the system.

In addition, actively controlled joint state responses have the following characteristics:

1) discontinuities - due to low frequency control loops,
2) non-linearities - due to joint state feedback, and
3) variations - due to constant control gains applied to changing system.

All of these characteristics contribute to a different joint response between actual dynamic manipulator systems and kinematically simulated manipulator systems. A standard resolved rate POR control scheme can effectively control the POR trajectory, but it can do little to control the joint trajectories. For a kinematically redundant manipulator, an infinite number of joint trajectories are possible for any given POR maneuver. Because of this, an actual space manipulator response (with a conventional resolved rate POR controller) is guaranteed to be different from a kinematic simulation response for the maneuver. Through a complex sequence of POR maneuvers (such as those proposed for Space Station assembly), the drift in the arm configuration between actual manipulator and simulated manipulator response will continue to grow through each successive maneuver potentially creating dangerous operational problems.

ANALYSIS OVERVIEW

To demonstrate the concepts and problems discussed above, two simulations runs are performed – one kinematic and one dynamic. A brief overview of the simulation math models is presented later. The kinematic simulation run consists of a POR maneuver, with acceleration and deceleration profiles, and an active position hold region. The dynamic simulation run consists of the exact same maneuver and position hold commands. The differences in the responses of the two simulations are discussed in detail to provide a better understanding of the control problems discussed above.

In addition to these runs, a second dynamic simulation is also performed incorporating POR force/torque control (instead of resolved rate control) during maneuver acceleration and deceleration regions. Results from this simulation are used to show advantages and disadvantages of stand-alone POR force/torque control.

Simulation Description

The kinematic simulation flow is depicted in Figure 1. The dynamic simulation flow is depicted in Figure 2. The manipulator guidance and control blocks of each simulation are *identical* to ensure proper comparison between the kinematic and dynamic simulations. With perfect sensing and perfect joint servos, the kinematic simulation is reduced to direct integration of the joint rate commands to produce joint positions; state integration and the control loop cycle times are both set to 80 milliseconds. The dynamic simulation, with perfect analog-to-digital conversion and perfect sensing, has a higher fidelity motor model including friction compensation logic for joint rate and torque commands, motor and joint friction, and joint gearbox backlash. As with the kinematic simulation, the manipulator guidance and control block for the dynamic simulation is also performed every 80 milliseconds. However, state integration is performed every 5 milliseconds with a fourth order modified midpoint integration scheme (3 step with 2 acceleration evaluations) [1].

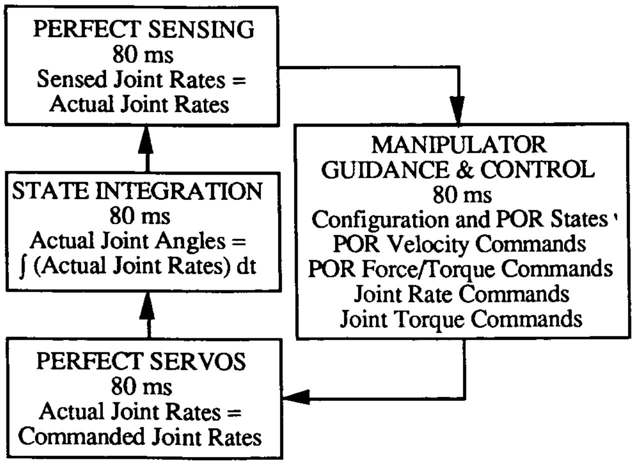

PERFECT SENSING
80 ms
Sensed Joint Rates = Actual Joint Rates
STATE INTEGRATION
80 ms
Actual Joint Angles = ∫ (Actual Joint Rates) dt
PERFECT SERVOS
80 ms
Actual Joint Rates = Commanded Joint Rates
MANIPULATOR GUIDANCE & CONTROL
80 ms
Configuration and POR States
POR Velocity Commands
POR Force/Torque Commands
Joint Rate Commands
Joint Torque Commands

Figure 1 - Kinematic Simulation Flow

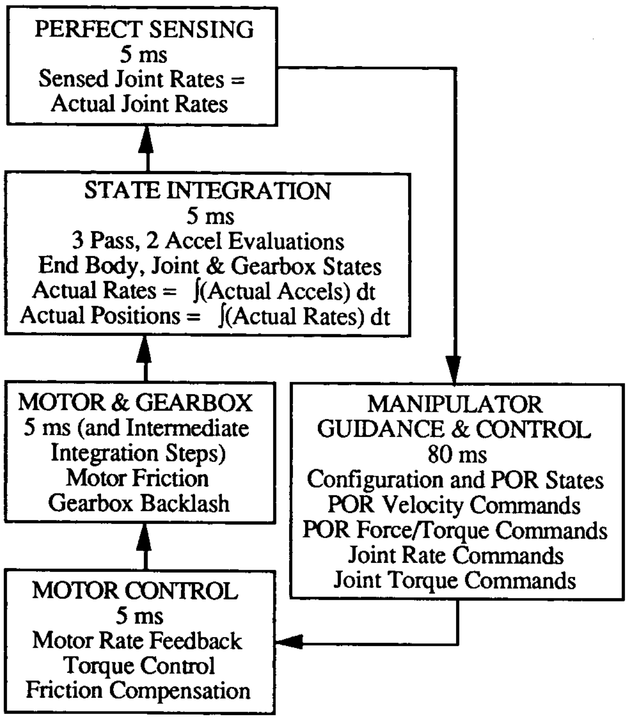

PERFECT SENSING
5 ms
Sensed Joint Rates = Actual Joint Rates
STATE INTEGRATION
5 ms
3 Pass, 2 Accel Evaluations
End Body, Joint & Gearbox States
Actual Rates = ∫(Actual Accels) dt
Actual Positions = ∫(Actual Rates) dt
MOTOR & GEARBOX
5 ms (and Intermediate Integration Steps)
Motor Friction
Gearbox Backlash
MOTOR CONTROL
5 ms
Motor Rate Feedback
Torque Control
Friction Compensation
MANIPULATOR GUIDANCE & CONTROL
80 ms
Configuration and POR States
POR Velocity Commands
POR Force/Torque Commands
Joint Rate Commands
Joint Torque Commands

Figure 2 - Dynamic Simulation Flow

The manipulator guidance and control block is shown in Figure 3. There are two important aspects of this diagram, POR rate control and POR force/torque control Rate control will be discussed first and then force/torque control.

For POR rate control, pointing vectors from the current POR states to the desired POR states are unitized and scaled by the POR maneuver rates to produce the POR rotational and translational commands. These commands are then scaled, depending on the current maneuver region, to produce the final POR rate commands. There are four distinct maneuver regions: acceleration, maneuver, deceleration, and active position hold. For each of these regions, POR rotation and translation commands are completely decoupled, i.e., POR rotation could be in the acceleration region while POR translation is in the active position hold region.

During the acceleration region, the POR rate command vectors are scaled based on elapsed time from zero to the maneuver rates thus emulating a constant acceleration profile. During the maneuver region, the commanded POR rates remain unchanged. During the deceleration and active position hold regions, the commanded POR rates are scaled from the maneuver rates to zero based on the "distance-to-go" to the desired POR end states; this function produces a parabolic velocity profile during braking. Using the final POR rate commands, joint rate commands are then generated via the resolved rate control law proposed by Whitney with constant unity weighting [2,3]. With the joint rate commands in hand, a gain for the friction compensation logic located in the motor model is calculated to provide a smooth transition for the friction compensation commands as the joint rates change direction. The motor model also contains a rate feedback loop.

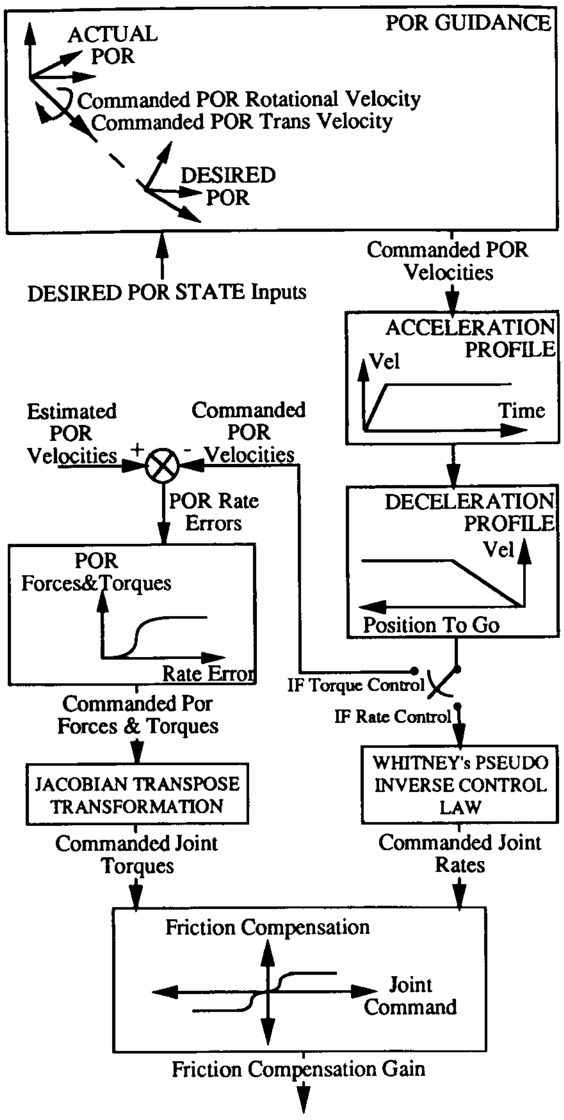


Figure 3 - Manipulator POR Guidance and Joint Rate/Torque Control

The POR force/torque control depends to a great extent on the existing POR rate control. The primary difference between the two control schemes is that during rate control, joint rate feeds back directly to the joint rate commands at a high frequency, but for force/torque control, the joint rate feedback is transformed to POR rate feedback and performed at a low frequency. Also, for the current analyses, the force/torque control is used only during the acceleration and deceleration maneuver regions. For these regions, the POR rates are fed back into the POR rate commands to produce POR rate error vectors. These vectors are unitized and then scaled from zero to the maximum maneuver forces and torques based upon the magnitude of the rate error vectors. These final commanded POR force and torque vectors are then transformed to joint torques via the transpose of the Jacobian matrix. Friction compensation gains again must be calculated to provide a smooth transition when the joint torque commands change directions.

The dynamic simulation motor model, presented in Figure 4, accepts both the joint rate and the joint torque commands. Torque commands are scaled directly to applied motor voltage (after gear reduction), while the rate commands go through gear reduction and are then differenced with the actual motor rates (rate feedback) before being scaled to applied motor voltages. Both sets of applied motor voltages are summed along with a friction compensation voltage that acts in the direction of the commanded joint rate (a simplistic description) to offset the effects of motor and joint friction. The resulting motor voltage is scaled to produce the applied motor torque. For this analysis, applied motor torque is calculated at 200 Hz (5 milliseconds).

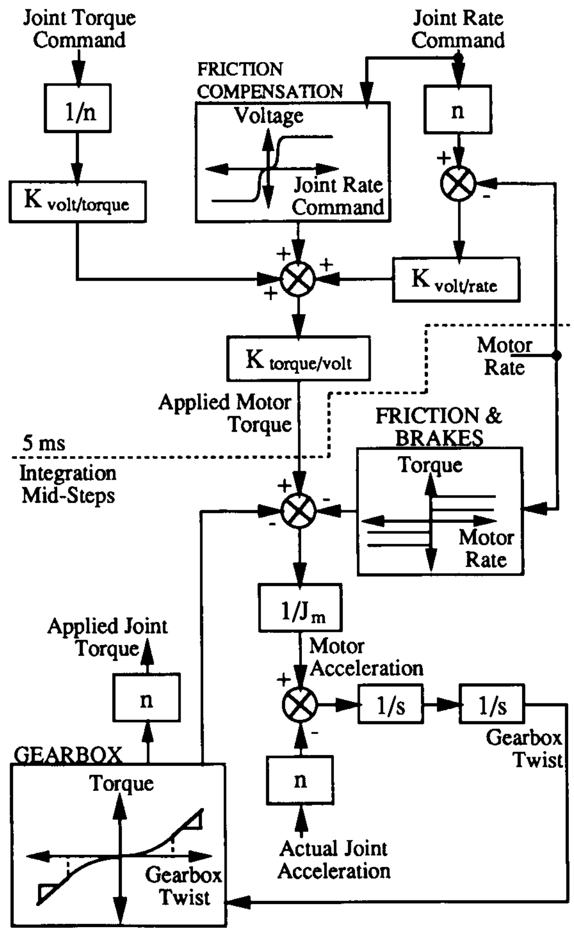


Figure 4 - Motor Rate/Torque Control and Gearbox Dynamics

Continuing with Figure 4, motor acceleration is determined by subtracting the torque due to motor friction and the gearbox torque from the applied motor torque, and dividing by the motor shaft inertia. This acceleration is differenced with the actual joint acceleration (after gear reduction) to produce the gearbox twist acceleration. The gearbox twist acceleration is then integrated twice to produce the gearbox twist angle used to determine the gearbox torque through a two stage gearbox backlash model. The first stage is non-linear to a breakout angle; the second stage is linear past the breakout angle. For the current analysis, gearbox twist state integration is performed over a 5 millisecond time step with the modified midpoint integration scheme; intermediate steps are synchronized with the joint state integration (from the manipulator dynamics).

Data Loads and Preparation

The manipulator system simulated consists of a Space Shuttle base vehicle (mass characteristics), a SPAS satellite payload (mass characteristics), and a Space Station RMS (mass and kinematic configuration). Data Loads for the simulations are taken primarily from published documents for the Space Shuttle, the Shuttle RMS, and the Space Station RMS. Mass Properties and attach point information for the Shuttle and the SPAS satellite payload are extracted from the PDRSS data book. Mass properties and kinematic configuration data for the SSRMS are extracted from a SPAR document. The motor, gearbox, and friction data for all 7 SSRMS joints is of the SRMS shoulder yaw joint, also from the PDRSS document. The torque to voltage and voltage to torque gains in the motor model also correspond to the SRMS shoulder yaw joint. Maneuver rates, tolerances, and breaking thresholds all correspond to specified data given for the SRMS manipulating the SPAS satellite. The maneuver selected for analysis is arbitrary with maneuver initial and final manipulator configurations depicted in Figures 5 and 6 respectively.

The motor model rate to voltage control gain was determined individually for all seven joints. First, brakes were applied on all joints. Next, brakes were relieved for a single joint and that joint was commanded to achieve a specified mid-range rotation rate. The rate to voltage gain was adjusted for the joint until the joint response yielded good acceleration and rate maintenance qualities. The initial configuration of the arm was identical for all control tests and significantly different from any of the configurations achieved during POR maneuver analysis.

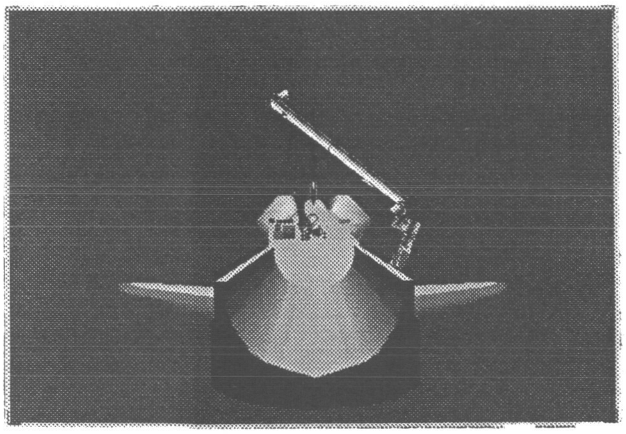


Figure 5 - POR Maneuver Initial Configuration

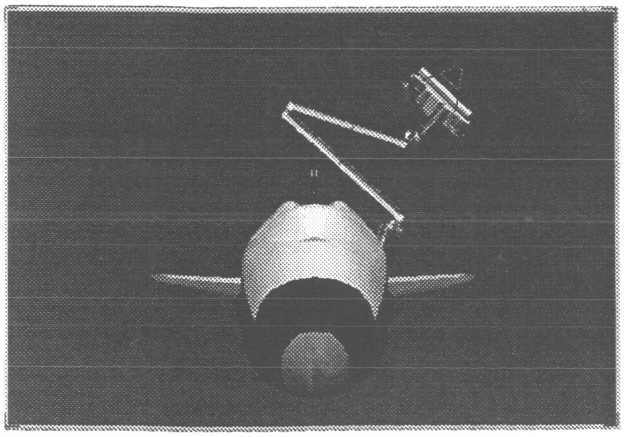


Figure 6 - POR Maneuver Final Configuration

RESULTS

Pertinent kinematic simulation results are presented in Figures 7 through 9. In Figure 7, notice the constant acceleration region (linear velocity profile), the constant maneuver rate region (constant velocity profile), the linear deceleration braking region (parabolic velocity profile), and the active position hold region (zero velocity profile). In Figure 8, the POR Euler attitude time histories represent a great arc rotation that can be more easily seen in the maneuver region of the rotational velocity plot. In Figure 9, the joint responses are smooth and continuous in the separate maneuver regions. In general, the kinematic simulation response is a "perfect" or ideal response.

Figures 10 through 12 represent analogous plots for the dynamic simulation. Notice that upon first glance the position and orientation histories appear almost identical and that the velocity profiles look very similar. This result demonstrates that fundamental controllability of the POR position and orientation is achievable. However, some important differences between the two simulation responses require additional discussion. To help visualize the actual differences between the two simulations, the POR translational vector data from kinematic simulation is subtracted from the dynamic simulation data. The magnitude histories of the resulting "difference", or error, is plotted in Figure 13.

The initial velocity error spike in Figure 13 demonstrates that the instantaneous velocities in the kinematic simulation can not be realistically achieved in a dynamic system. This initial velocity error is the major contributor to the overall POR position error which is close to four inches for this maneuver. However, as shown by the joint angle error histories in Figure 14, the largest arm configuration error occurs during the maneuver region which demonstrates the arm's tendency to drift from the expected configuration.

Returning to Figure 13, small perturbations can be seen during the maneuver region (up to 75 seconds). These perturbations are caused by the most serious control problem we faced while performing these analyses: friction. Compare perturbation time slots in Figure 13 with the time slots in Figure 12 where the joint angle histories change direction, i.e., when the joint velocities become zero. These regions are dominated by joint friction. The velocity perturbations in these regions are caused when 1) the joint initially stops due to friction, and 2) when the joint overcomes friction and breaks loose. Both instances cause discontinuities in the arm motion effectively reducing the system degrees of freedom to something less than seven; this will create problems when the control system expects seven full degrees of freedom to be available for control. Friction problems prompted us to develop the friction compensation logic (only touched upon in this paper) greatly improving our overall arm response (as is evident with the four inch maximum path deviation). We also believe these perturbations can be significantly reduced by placing the low frequency portion of the friction compensation logic with the high frequency portion of the logic.

The second velocity spike of Figure 13 occurs at the beginning of the deceleration region, much like the first spike. The secondary spikes occurring around 90 seconds are caused by a control mode change between the deceleration and active position hold regions. This mode change is primarily a friction compensation gain logic change to allow a finer control for position hold. Notice that the final positional error is less than half an inch.

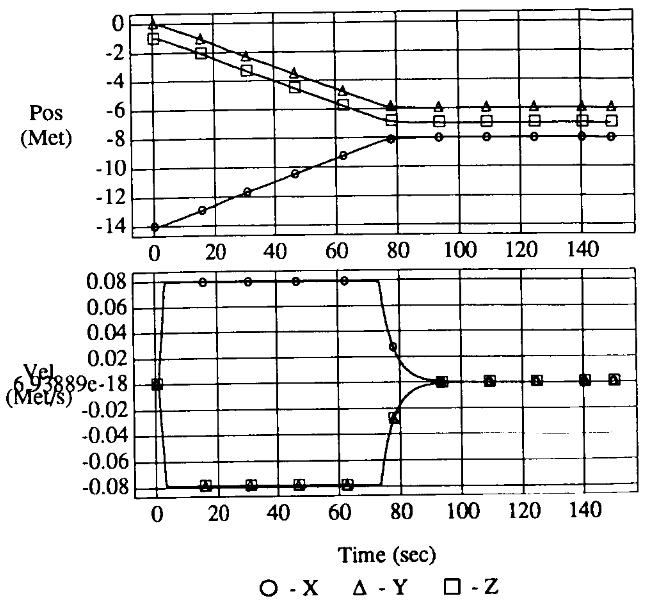


Figure 7 - Kinematic Simulation POR Position and Translational Velocity Histories

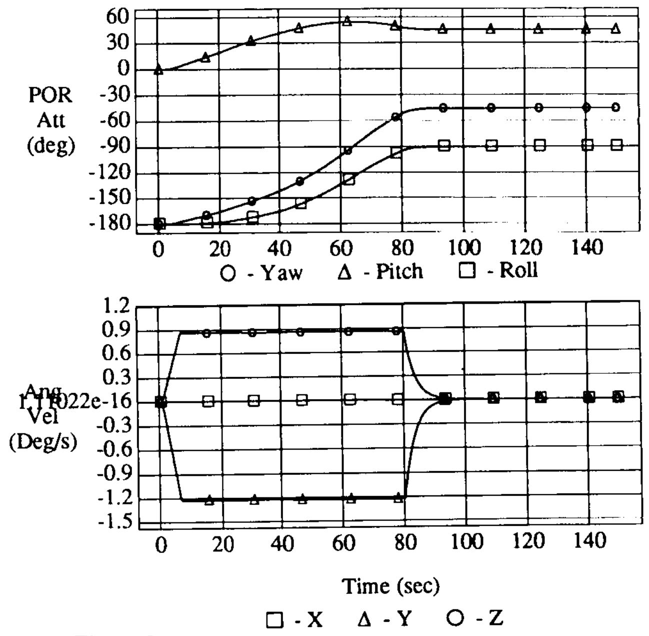


Figure 8 - Kinematic Simulation POR Orientation and Rotational Velocity Histories

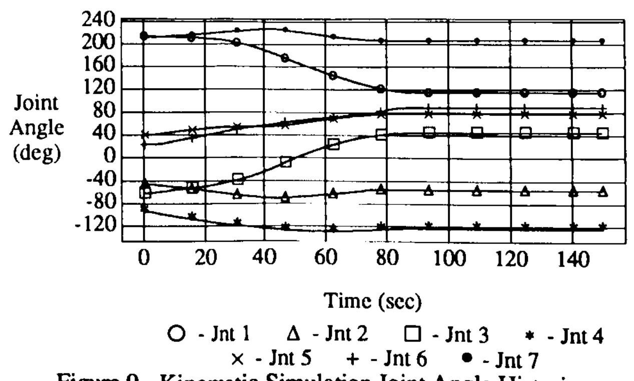


Figure 9 - Kinematic Simulation Joint Angle Histories

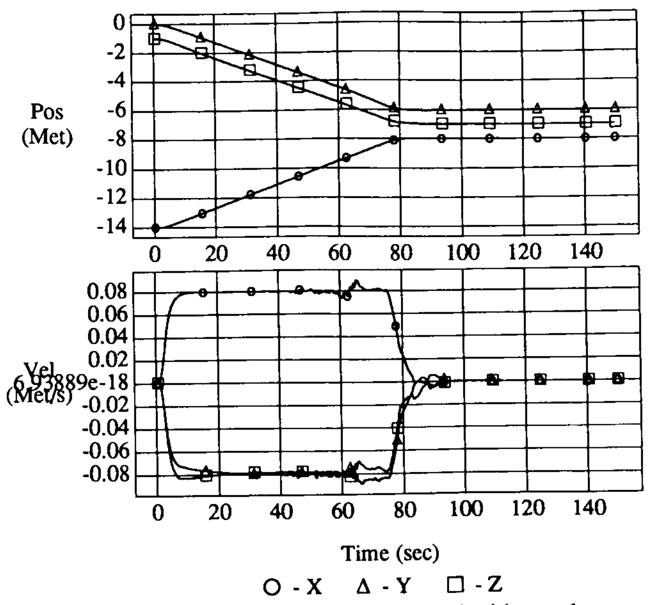


Figure 10 - Dynamic Simulation POR Position and Translational Velocity Histories

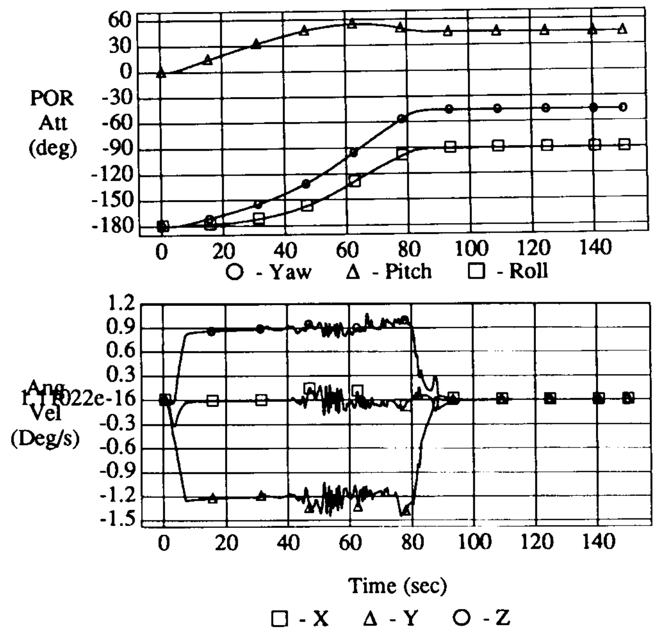


Figure 11 - Dynamic Simulation POR Orientation and Rotational Velocity Histories

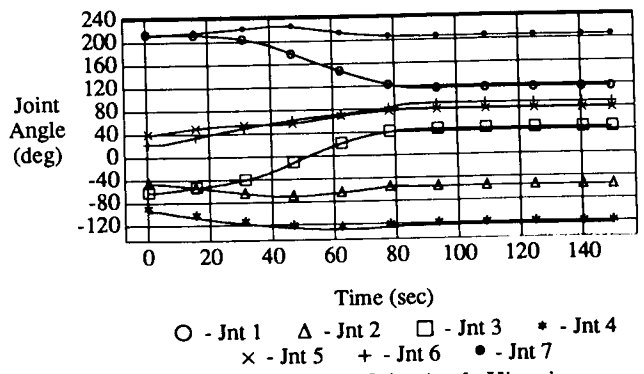
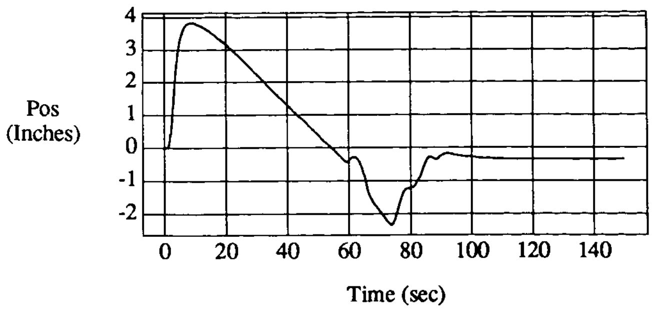


Figure 12 - Dynamic Simulation Joint Angle Histories



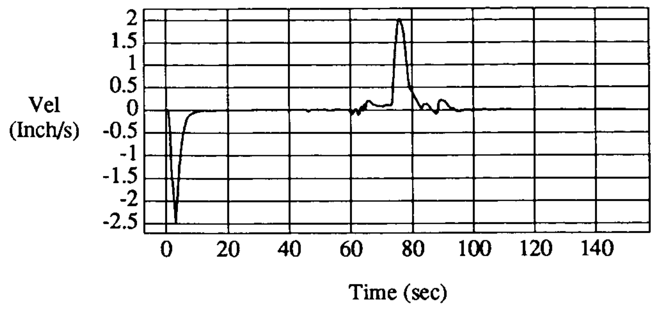
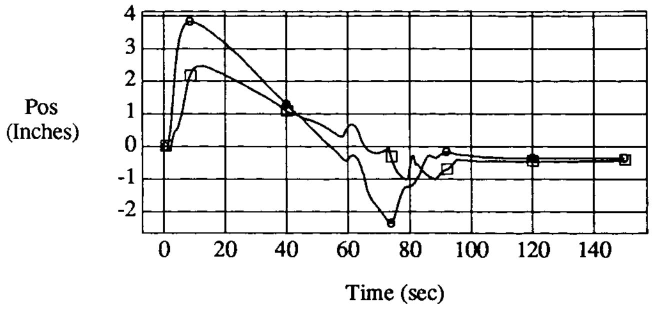
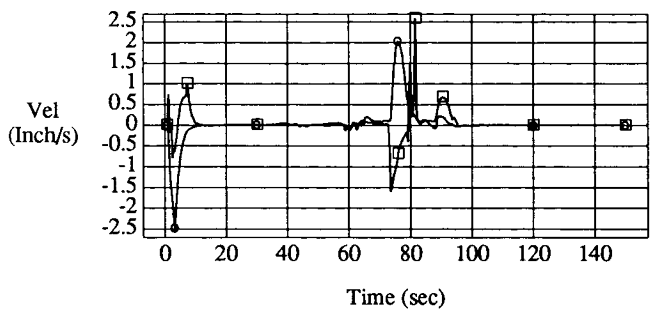


Figure 13 - POR Position and Translational Velocity Difference (Dynamic Simulation - Kinematic Simulation)





O - Nominal Control □ - POR Force/Torque Control

Figure 15 - POR Force/Torque Control Comparison

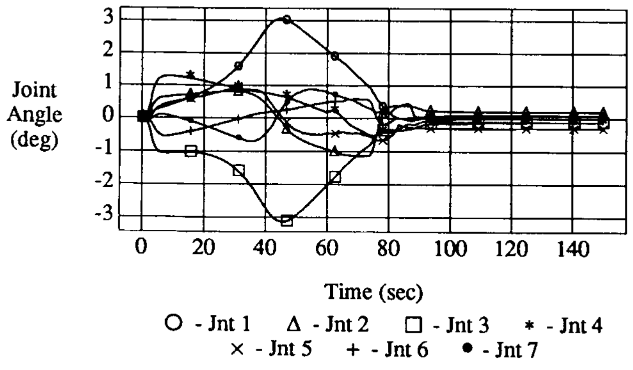


Figure 14 - Joint Angle Difference History

Substituting the POR force and torque control mode during the acceleration and deceleration regions produces interesting results. Figure 15 shows Figure 13 with overplots of the POR force/torque control simulation. Notice how the force/torque control responds much faster to the initial rate command than does the rate control (0 to 5 second region). Also notice the same to be true during the deceleration region (75 to 90 seconds). The most interesting aspect of the force/torque overplot is the large velocity spikes near 80 seconds. These spikes are again caused by joint friction and they are much more pronounced than the friction spikes of the rate control regions. This can be attributed to the difference in rate feedback frequencies; rate feedback for the force/torque control occurs at 12.5 Hz whereas the rate feedback for the rate control occurs at 200 Hz. Obviously, the rate controller will be able to react much better to discontinuities than the force/torque controller. However, even with the large velocity spikes the addition of the POR force/torque control improved the overall path deviation throughout the maneuver.

CONCLUSIONS

POR rate control is successfully achieved for kinematically redundant space manipulators and POR maneuvers are generally repeatable. In task space, there is good agreement between kinematic and dynamic simulations. However, arm configurations through the maneuvers are generally not repeatable between kinematic and dynamic simulations which suggests a drawback to complicated task scenario development using kinematic simulations.

The primary goal of most manipulator task scenario development is to reach the successive POR positions and orientation with a benign (no collisions, no singularities, etc.) arm configuration. If the arm configuration is not precisely predictable with a kinematic simulation then one of two events needs to happen. Either analyses should be performed with a reasonably high fidelity dynamic simulation, or a kinematic controller needs to be developed which controls both manipulator task space and configuration space motions.

Although the force/torque controller improved the POR response, the arm configuration drift problem still exists. We believe this problem will never be completely solved until some type of hybrid POR and arm configuration controller is developed to control the task space and configuration space aspects of the problem concurrently. Perhaps an adaptive controller utilizing varying system dynamic characteristics in conjunction with the POR force and torque control principles presented here, or possibly a 6+1 degree-of-freedom (DOF) controller which controls a single joint independently of the others to "fix" a 6 DOF solution for POR rate control.

We also believe that the POR force and torque control scheme presented here can be enhanced to provide rate control equivalent control during the maneuver region. The advantage of this type of controller is that no mathematical singularities exist. The transformation between POR forces and torques and joint torques is the transpose of the Jacobian matrix, a matrix which does *not have to be inverted* and thus will not exhibit control singularities.

REFERENCES

[1] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, Massachusetts, 1988, pp. 580 - 582 .

[2] Whitney, D., "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems*, Vol. MMS-10, No. 2, 1969, pp. 47 - 53.

[3] Bailey, R., Quiocho, L, and Cleghorn, T., "Kinematically Redundant Arm Formulations for Coordinated Multiple Arm Applications," *Proceedings of the Third Annual Workshop on Space Operations Automation and Robotics*, 1989, pp. 447 -454.

[4] "SRMS Master Parameters List", SPAR-R.775 H, February 1983.

[5] "PDRS Database", NASA/JSC Letter VP-88-058, March 1988.

[6] "Space Shuttle Operation Level C Functional Subsystem Software Requirements Document Remote Manipulator System (RMS)", STS 87-0017, November 1987.

N91-20643

IVA THE ROBOT: DESIGN GUIDELINES AND LESSONS LEARNED FROM THE FIRST SPACE STATION LABORATORY MANIPULATION SYSTEM

Carl R. Konkel
Allen K. Powers
J. Russell Dewitt

Teledyne Brown Engineering
300 Sparkman Drive
Huntsville, Alabama 35807

ABSTRACT

The first interactive Space Station Freedom (SSF) laboratory robot exhibit has been installed at the Space and Rocket Center in Huntsville, Alabama and has been running daily since March. IVA the Robot is mounted in a full scale United States Laboratory (USL) mockup to educate the public on possible automation and robotic applications aboard the SSF. Responding to audio and video instructions at the Command Console, exhibit patrons may prompt IVA to perform a housekeeping task or give a speaking tour of the module. Other exemplary space station tasks are simulated and the public can even challenge IVA to a game of tic-tac-toe.
In anticipation of such a system being built for the Space Station, a discussion is provided of the approach taken, along with suggestions for applicability to the Space Station environment.

INTRODUCTION

Teledyne Brown Engineering is pursuing an IR&D effort in robotics and automation in support of the NASA Space Station Freedom. This effort was initiated by user requirements which underscore the need for enhanced levels of automation and machine intelligence to reduce crew workloads. SSF phase B studies have shown available crew time as one of the most limited consumables and an obvious solution would be to automate those repetitive tasks which lend themselves to automation. This would allow the crew to perform the more sophisticated duties which require human dexterity and interaction. A prototype of an IntraVehicular Activities (IVA) robot for a conceptual Laboratory Manipulator System has been assembled as IVA the Robot (Figure 1) within a Space Station Freedom mockup at the Space and Rocket Center in Huntsville, Alabama.

IVA THE ROBOT

The Space and Rocket Center is a functional museum which presents a unique history of America's adventures in space as well as conceptual models of what may be expected in future space efforts. The intent of a Laboratory Manipulator System within the Space Station mockup is to display realistic space robotic applications to a public which expects C3PO and R2D2 capabilities from robotic systems. Following is a discussion of original design requirements and the approach taken to fulfill those specifications.

Design Requirements: The original IVA the Robot Exhibit specification detailed the location, weight, dimensions, configuration and appearance of the Laboratory Manipulator System. IVA was to be located on a six-foot track suspended from the laboratory module and be able to reach three racks of equipment. Standard Space Station tasks such as picking up a cylindrical sample and placing it in a materials processing furnace, or opening a door to remove a piece of equipment were specified. Among the more specific capabilities was the definition of a "wipe-down" task in which the robot would open a storage door, remove a sponge, proceed to clean a wall, and replace the sponge. Another unique task, included for both entertainment value as well as dexterity demonstration reasons was a Tic-Tac-Toe game played between the robot and an observer. Figure 2 is a block diagram of IVA's major elements and their interfaces.
All of the required tasks are initiated from the Control Console by the user. Based on user inputs from a joystick or pushbuttons, IVA will perform the following tasks:

1) Housekeeping. This task has already been defined for the SSF habitable modules and crew members were the first to suggest a "cleaning robot" to do it. IVA removes a sponge from a storage compartment, wipes down the walls and replaces the sponge in the compartment. Concurrently, a voice system is used to explain what IVA is doing and why it is necessary.

2) Furnace sample changeout. In an effort to demonstrate the usefulness of a robot for experiment manipulation, a task to changeout samples in a materials processing furnace was defined. First, IVA opens a storage door within which samples are stored. Then IVA proceeds to the furnace where the "heated" sample is removed and placed in an empty storage bin. A "fresh" sample is selected from the storage bin and placed into the furnace and all doors are closed. Sensors are located in the furnace compartment and in the storage bin to sense the location of all samples. Intelligence is programmed into IVA so that samples are not placed into an occupied slot. Again, IVA speaks to the exhibit user and explains the process.

3) Tours of the Space Station and module. There are two tours which have IVA speaking to the public explaining the Space Station, the Laboratory Module, and IVA's purpose. During the discussion, IVA guides the user by pointing and motioning to the various items of interest.

4) Manual Mode. To give the public a direct feel of operating a robot, IVA responds to various joystick and push button inputs which give the user direct control of the robot. Precautions are programmed so that IVA can't collide with anything or damage laboratory components.

5) Tic-tac-toe. One of IVA's more ambitious programming tasks, tic-tac-toe demonstrates IVA's flexibility and dexterity in a fun way to exhibit patrons. The game is controlled by the user at the console monitor and the game pieces are physically moved by IVA. This is the most popular task and IVA isn't easy to beat. On the average against a good player, IVA will win 20% of the time, tie 60% of the time and lose the rest. As with all tasks, IVA uses the voice system to explain the game and how to operate the controls.

IVA is powered up around the clock and runs in three basic modes. The first mode includes all the interactive actions described above, where IVA interacts directly with the public. Another mode is the non-interactive mode in which IVA will run different tasks from those discussed above until someone approaches the command console. There is a pressure mat in front of the command console so that IVA knows when someone is at the controls. The last mode is a "sleep" mode where IVA will park and go into a dormant state when the museum is closed for the night.

An integrated system, IVA is built mostly from off the shelf components. The brains of the system is an IBM PC/AT using the GWBASIC and Quick Basic programming languages. The PC reads inputs from exhibit patrons at the command console and converts those inputs to directives to the robot arm and exhibit peripherals. The PC also schedules events like the parking of the robot at night, occasional recalibrations and keeps records of all tasks performed by IVA. Given a quantitative record of activity, the exhibit may later be streamlined according to usage. Having a written record also helps the troubleshooter to pinpoint problems by showing which routine was executing and when a problem occurred. The voice capabilities are provided with a COVOX VOICEMASTER system. This speech system digitizes a spoken message and allows the ability to prompt IVA's speech within the control program. The robot arm is a Canadian-made CRS M1 six axis arm mounted on a six foot track bolted to the mockup ceiling. Programs which move the robot gripper to teach points are written in the CRS particular language and are prompted in response to signals from the PC. Lastly, a control/SVC panel was built to house the OPTO22 I/O boards. This panel distributes signals between the robot and the PC, the robot and the exhibit, and the PC and the exhibit.

IVA has demonstrated the ability of a single 6 axis track-mounted robot to successfully service three racks within the Space Station exhibit. But before any robotic system will be implemented within Space Station Freedom, crew and lab safety issues must be addressed and solved. Regardless of obvious workload benefits from a robotic system, there must be complete assurance that no detriment to the crew or lab hardware can occur at the hands of that system. The logistics of being a museum exhibit precludes exhibit patrons from any direct physical harm from IVA. The complete safety system is a "deadman" switch located in the access door to IVA's module, so entering the workspace powers down the robot. Certainly, this isn't possible in a Space Station environment where it is conceivable that man and robot may directly interact, therefore demanding the demonstration of harmonious interaction between the two.

Lessons Learned. Tho onboard IVA robot element of the system has been working successfully on a daily 12-hour basis. Minor problems such as workspace incompatibilities with the original teach points and robot voice coordination were anticipated and solved early. Unanticipated problems included intermittent failure of flexible cable wires exhibiting symptoms analogous to "software" errors, and early saturation of I/O signals.

CONCLUSION

The IVA the Robot exhibit was developed with the anticipation of the implementation of robotic systems within Space Station Freedom. The exhibit has been operational since March, 1990 and IVA services three racks of mock equipment. Extension of an IVA-type robot to SSF could free astronauts from repetitive tasks for better usage of their time. Such a system may be directly voice controlled by an astronaut or possibly even remotely controlled by a ground station operator. In an effort to alleviate astronaut workload, it is believed that while classical robotic techniques are readily applicable to specific tasks, the major area of concern is to crew and laboratory safety.

ACKNOWLEDGEMENTS

The authors wish to recognize Mike Cobb, the electronics techician for this project. With a keen insight of electronic troubleshooting, Mike considerably shortened the Lessons Learned portion of this paper, saving valuable time and a portion of the authors' sanity.

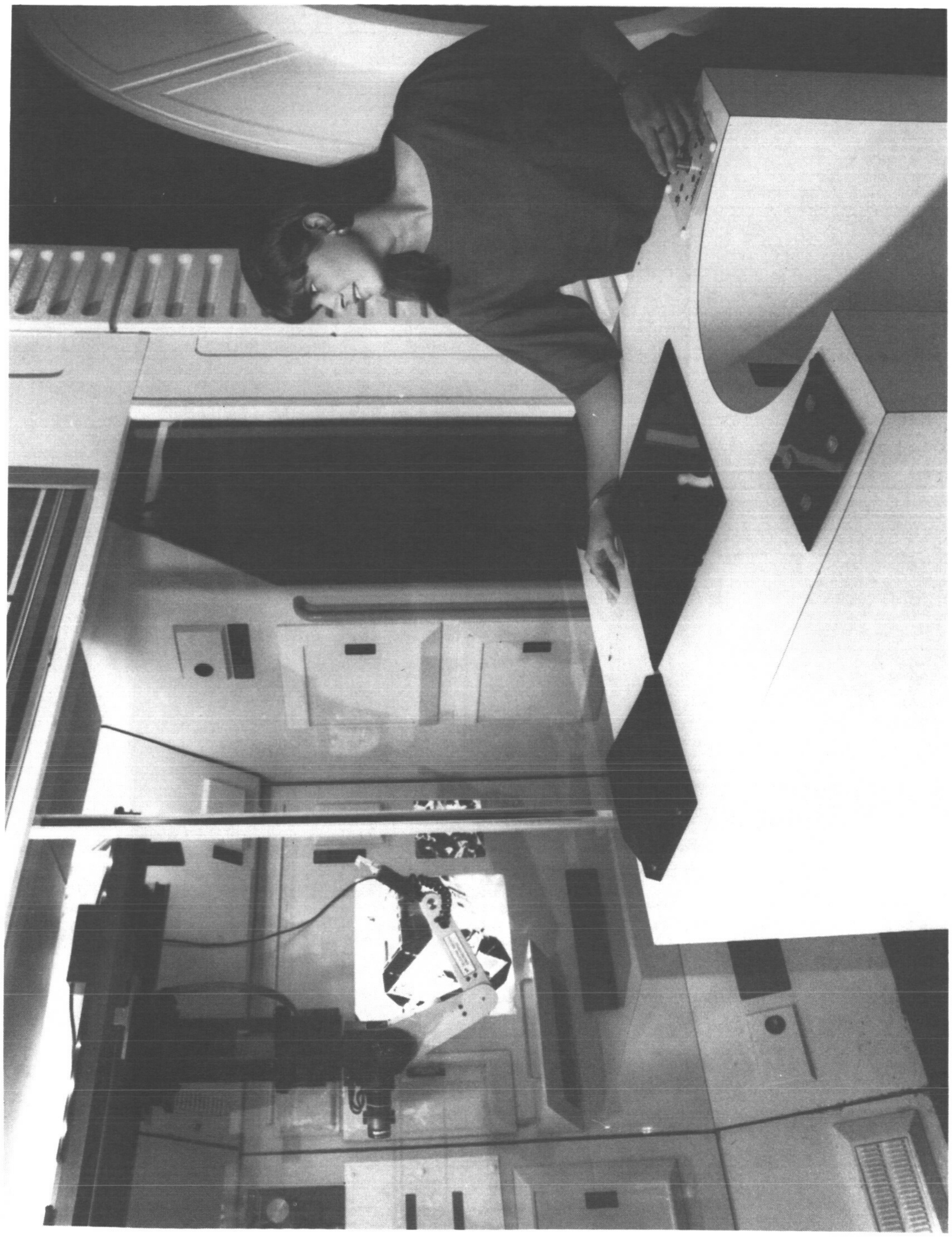


Figure 1. IVA the Robot

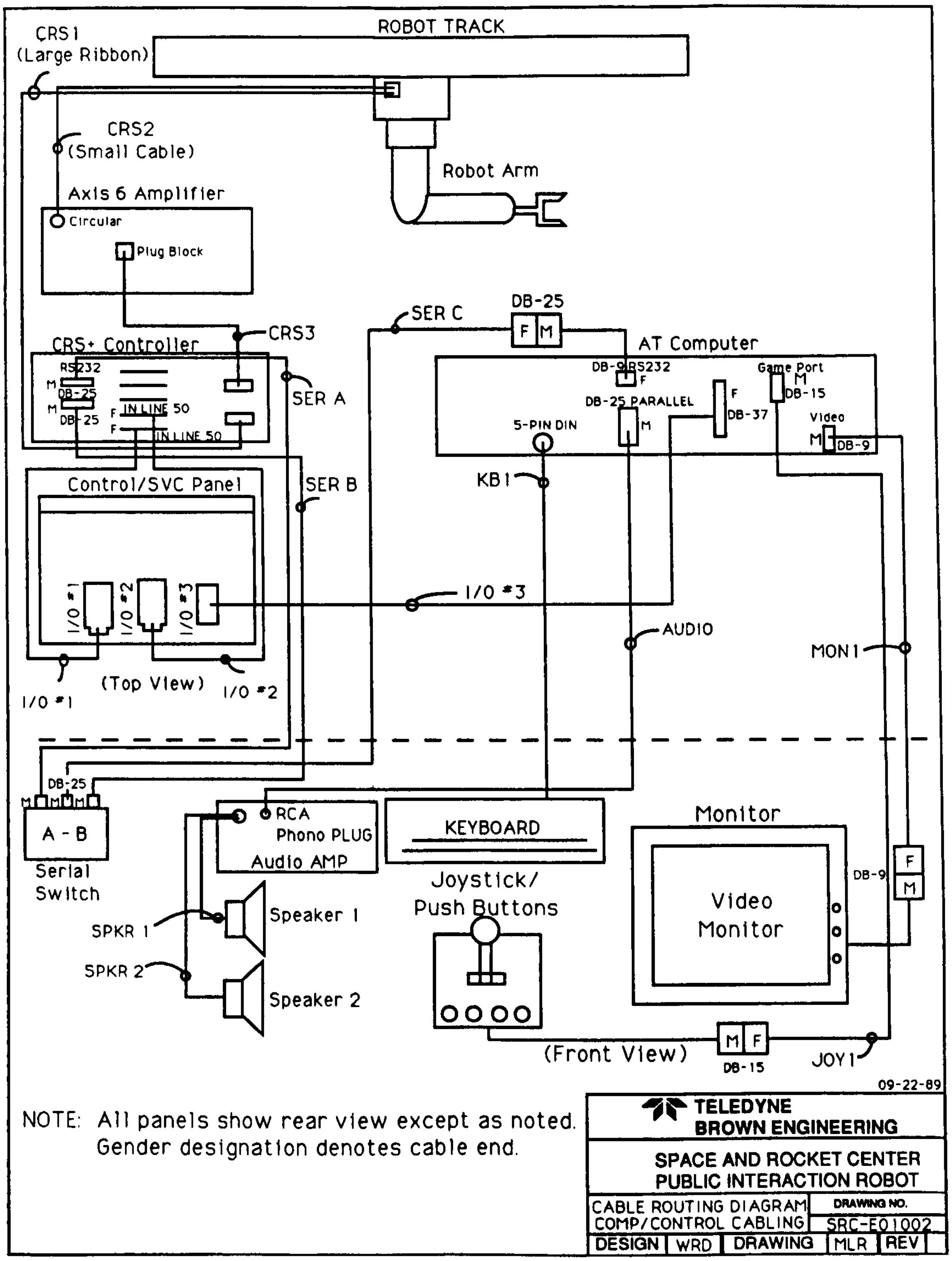


Figure 2. Exhibit elements and interfaces

N91-20644

A NOVEL DESIGN FOR A HYBRID SPACE MANIPULATOR

M. Shahinpoor

Department of Mechanical Engineering

University of New Mexico

Albuquerque, New Mexico 87131

ABSTRACT:

Described are the structural design, kinematics and characteristics of a novel robotic manipulator for space applications and, in particular, utilization as an articulate and powerful space shuttle manipulator. Hybrid manipulators are parallel–serial connection robots that give rise to a multitude of highly articulate robot manipulators. These manipulators are modular and can be extended by additional modules over large distances. Every module has a hemi–spherical work space and collective modules give rise to highly dexterous symmetrical work space.In this paper some basic designs and kinematical structures of these robot manipulators are discussed, the associated direct and the inverse kinematics formulations are presented, and solutions to the inverse kinematic problem are obtained explicitly and elaborated upon.

These robot manipulators are shown to have a strength–to–weight ratio that is many times larger than the value that is currently available with industrial or research manipulators. This is due to the fact that these hybrid manipulators are stress–compensated and have an ultra light weight, yet, they are extremely stiff due to the fact that the force distribution in their structure is mostly axial. The means of actuation in these manipulators are entirely prismatic and can be provided by ball–screws with anti–backlash nuts for maximum precision.

INTRODUCTION

Serially connected robot manipulators in the form of an open–loop kinematic chain with computer–controlled joint actuation have been examined extensively in the robot engineering literature (see Shahinpoor [1]). These examinations study the structural design, kinematics, dynamics, trajectory planning, work space design, control and stability. On the other hand the pertinent literature on parallel–connection robot manipulators is scarce as discussed by Fichter [2][3]. A classic example of a parallel manipulator is the Stewart platform (see Stewart [4]) which has been kinematically and to some extent dynamically investigated by Fichter [2]. Other similar mechanisms and manipulators have been discussed by Earl and Rooney [5], Hunt [6], and Yang and Lee [7].

In the present paper we introduce yet another novel robotic structure of a hybrid nature. In these hybrid manupulators both serial elements and parallel elements are present and can be actuated in a prismatic fashion to give rise to a highly articulate robot manipulator with hemispherical work space and complete symmetry of movements within its work space. Figure 1 illustrates such a hybrid robot manipulator. Note that this structure particularly relates to a computer–controlled robotic arm capable of moving three dimensionally and symmetrically throughout its hemispherical workspace.

Computer–controlled robotic arms have been extensively used throughout the world and particularly in the US and Japan. See Shahinpoor [1] for a comprehensive literature survey on various kinds of robot manipulators and structural designs. Two basic problems have been associated with conventional robot manipulators as described below:

1– They are generally made massive and stiff so as to
eliminate motion control problems associated with
structural flexibility.

2– They generally move slow because of the fact that they
are made massive and faily rigid.

Thus, there has been a great need in the manufacturing industry, government laboratories as well as defense organizations to develop light–weight, stiff and subsequently fast moving robot manipulators. The structure shown in Figure 1 and described in the following section achieves the above objectives and corrects for the above deficiencies of the conventional robot manipulators. Since all of its legs are simply supported at both ends by three dimensional joints such as universal or ball–and–socket joints the stresses in them are only axially distributed and thus give rise to a stress–compensated robotic structures. The structure shown in Figure 1 also has a minimum amount of extra mass and is essentially an ultra–light weight manipulator. Thus, it provides an ultra–light weight, stress–compensated robotic arm capable of fast motions.
It is further capable of moving symmetrically and hemi–spherically about its base platform ; something that most current robotic structures are unable to do.

In accordance with the present paper we describe a 7 degree of freedom robotic arm comprising a three–dimensional universal joint and two segments of a robotic arm such that the one end of the first segment is fixed to a base platform in the form of an equilateral triangular structure with the other end attaced to a joint platform which is another equilateral triangular structure. The one end of the second segment is attached to the joint platform with the other end attached to a gripper platform which is another equilateral triangular structure such that it is basically free to move but other–wise equipped with a robotic hand, gripper, end–effector or fixture. The base platform is comprised of an equilateral triangle whose sides are made from metallic or otherwise strong material. The said joint platform is comprised of another equilateral triangular structure with strong sides positioned opposite to the base platform such that the vertices of the base platform and the joint platform are connected by means of a triplet of criss–crossed woven or single wires with a movable joint. The gripper platform is also an equilateral trianglular structure positioned oppositely to the sides of the joint platform such that the vertices are connected oppositely to the vertices of the joint platform first by a pair of criss–crossed or woven wires and also to the middle points of the sides of the joint platform by means of a set of linear actuator. The end–effector which may be a robotic gripper is attached via a set of support bars to yet another equilateral triangle, namely, an extended gripper platform with sides and vertices oppositely oriented with respect to the gripper platform. The gripper action may be provided by an intermediate mechanism.

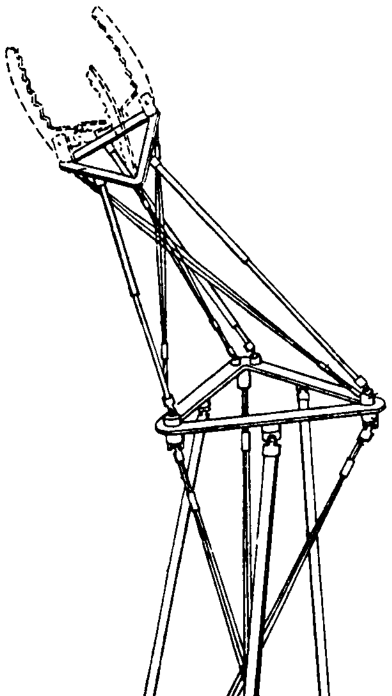


Figure 1– A platform structure of a hybrid robot manipulator

The actuation is provided by a set of six linear actuators . These linear actuators are such that three of them connect the vertices of the base platform to the mid section of the sides of the joint platform. Subsequently the other three linear actuators connect the mid section of the sides the joint platform to the vertices of the gripper platform.

The linear actuation may be hydraulic, pneumatic or electromagnetic. In case of hydraulic or pneumatic actuation the fluid motion control is provided by either digital or analog controllers comprising of electromagnetic valves. In case of electromagnetic actuation the linear actuators may be magnetic–induction or magnetic–coil driven or comprised of motorized ball screws for linear actuation. The gripper may also be actuated either hydraulically, pneumatically or electromagnetically. Due to the fact that the support bars create a kinematically constrained motion for the platforms the linear motion of the actuators must be performed in harmony so as not to violate the kinematical constraints. Here below a complete kinematic description of this robot manipulator is presented. This kinematical modeling is necessary for computer–controlled motion of the robotic gripper.

The fundamental question answered here is :

"Given the desired location and orientation of the gripper in the hemi–spherical work space of the manipulator what are the six values of the linear displacements of the 6 actuators in order to place the said gripper correctly at the desired position and with the desired orientation."

Let us refer now to Figure 2 which depicts a kinematic embodiment of the invention.

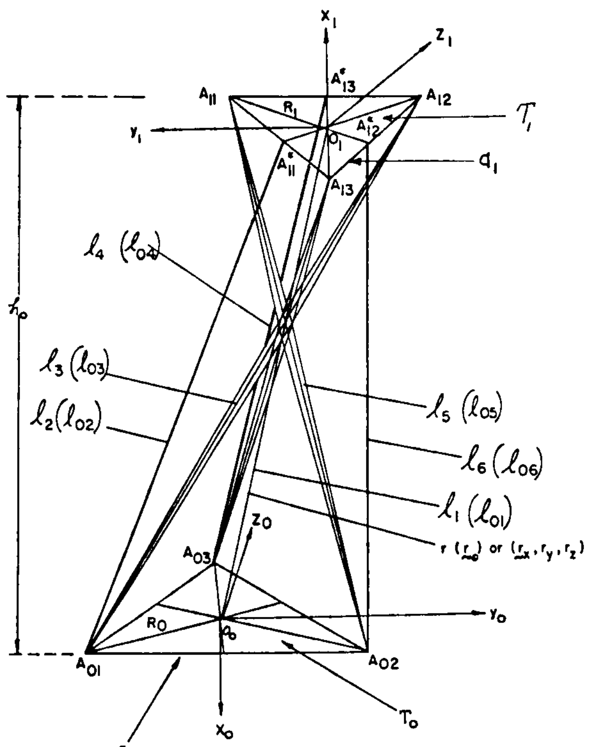


Figure 2– The kinematical structure of a 3–axis hybrid manipulator.

Note that, the present an analytical representation of the kinematic of the hybrid manipulator, a fixed reference rectangular cartesian frame is assumed with its origin at the point of intersection of angle bisectors or the medians of the said base platform, which is , hereon, called T_o A corresponding rectangular cartesian frame x, y, z, is considered fixed to the center of the said joint platform which is called T_1 .

The locations of points A_{o1}, A_{o2}, A_{o3}, A_{11} A_{12} and A_{13} are given by

$$R_{A_{o1}} = ((\sqrt{3}/6)a_o, -(a_o/2), 0)^T \quad (1)$$

$$R_{A_{o2}} = ((\sqrt{3}/6)a_o, (1/2)a_o, 0)^T \quad (2)$$

$$R_{A_{o3}} = ((-\sqrt{3}/3)a_o, 0 , 0)^T \quad (3)$$

with respect to the base frame T_o and by

$$R_{A_{11}} = ((+\sqrt{3}/6)a_1, +(1/2)a_1, 0)^T \quad (4)$$

$$R_{A_{12}} = ((3/6)a_1, -(1/2)a_1, 0)^T \quad (5)$$

$$R_{A_{13}} = ((-\sqrt{3}/6)a_1, 0, 0)^T \quad (6)$$

with respect to the platform frame T_1.

Consider an equilibrium reference position of the upper triangle T_1 with respect to the lower triangle T_o such that they are parallel with a perpendicular separation of h_o for which all lengths l_1 through l_6 are equal to l_1 through l_6. Under these circumstances the position of O_1 the origin of the frame T_1 with respect to T_o is given by a vector $\underset{\sim}{r}_o$ which is, however, generally $\underset{\sim}{r}$.

In the reference configuration the coordinate frame T_1 can be expressed with respect to the frame T_o by means of a 4x4 homogeneous transformation

$$[T_1]_o = \begin{bmatrix} -1 & 0 & 0 & -b_o \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Now let the origin of the T_1 frame in the upper said joint platform move to an orbitrary position $\underset{\sim}{r} = (r_x, \underset{\sim}{r}_y, r_z)^T$ and orientation θ, φ, ψ, such that θ, φ, and ψ are the corresponding angles in a right–handed fashion, between the pairs of axes (x_o, x_1), (y_o, y_1), and (z_o, z_1), respectively. In this arbitrary position and orientation the frame T_1 can be expressed with respect to the frame T_o by means of another 4x4 homogeneous transformation $[T_1]$ such that

$$[T_1] = \begin{bmatrix} \cos\theta & \cos(x_1,y_o) & \cos(x_1,z_o) & r_x \\ \cos(y_1,x_o) & \cos\ \varphi & \cos(y_1,z) & r_y \\ \cos(z_1,x_o) & \cos(z_1,y_o) & \cos\ \psi & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (8)$$

or

$$[T_1] = \begin{bmatrix} d_{11} & d_{12} & d_{13} & r_x \\ d_{21} & d_{22} & d_{23} & r_y \\ d_{31} & d_{32} & d_{33} & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

where d_{ij}, i,j = 1,2,3 are the direction cosines between the T_o and the T_1 frames, i.e.,

$$d_{ij} = \mathrm{Cos}\ (x_{1i}, x_{jo}), \quad (10)$$

$$x_{1i} = (x_1, y_1, z_1)^T, \quad (11)$$

$$(x_{jo}) = (x_o, y_o, z_o)^T \quad (12)$$

Thus, the location of all points on the upper triangle can be obtained with respect to the T_1 frame such that

$$0_1 \rightarrow \quad r_{o_1} = (0,0,0)^T \quad (13)$$

$$A_{11} \rightarrow \quad R_{A_{11}} = ((\sqrt{3}/6)a_1, (1/2)a_1, 0)^T \quad (14)$$

$$A_{12} \rightarrow \quad R_{A_{12}} = (\sqrt{3}/6)a_1, -(1/2)a_1, 0)^T \quad (15)$$

$$A_{13} \rightarrow \quad R_{A_{13}} (-(\sqrt{3}/3)a_1, 0, 0)^T \quad (16)$$

$$A^*_{11} \rightarrow \quad R^*_{A_{11}} (-(\sqrt{3}/12)a_1, (1/4)a_1, 0)^T \quad (17)$$

$$A^*_{12} \rightarrow \quad R^*_{A_{12}} (-(\sqrt{3}/12)a_1, -(1/4)a_1, 0)^T \quad (18)$$

$$A^*_{13} \rightarrow \quad R^*_{A_{13}} ((\sqrt{3}/6)a_1, 0, 0)^T \quad (19)$$

and

$$r_{A_{11}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{\underset{\sim}{A}_{11}} = [T]\,\underset{\sim}{R}^{(H)}_{A_{11}} = [T]\begin{bmatrix}(\sqrt{3}/6)a_1\\ (1/2)a_1\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}(\sqrt{3}/6)a_1d_{11}+(1/2)a_1d_{12}+r_x\\ (\sqrt{3}/6)a_1d_{21}+(1/2)a_1d_{22}+r_y\\ (\sqrt{3}/6)a_1d_{31}+(1/2)a_1d_{32}+r_z\\ 1\end{bmatrix} \qquad (20)$$

where $R^{(H)}_{A_{11}}$ is the homogeneous representation of $\underset{\sim}{R}_{A_{11}}$. and Similary

$$r_{A_{12}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{A_{12}} = [T]\mathbf{R}^{(H)}_{A\ 12} = \begin{bmatrix}(\sqrt{3}/6)a\\ -(1/2)a_1\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}d_{11}\ \ (3/6)a_1-(1/2)a_1d_{12}+r_x\\ (\sqrt{3}/6)a_1d_{21}-(1/2)a_1d_{22}+r_y\\ (\sqrt{3}/6)a_1d_{31}-(1/2)a_1d_{32}+r_z\\ 1\end{bmatrix} \qquad (21)$$

$$r_{A_{13}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{A_{13}} = [T_1]\,\mathbf{R}^{\mathbf{H}}_{A_{13}} = [T_1]\begin{bmatrix}-(\sqrt{3}/3)a_1\\ 0\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}-(\sqrt{3}/3)a_1d_{11}+r_x\\ -(\sqrt{3}/3)a_1d_{21}+r_y\\ -(\sqrt{3}/3)a_1d_{31}+r_z\\ 1\end{bmatrix} \qquad (22)$$

$$r^*_{A_{11}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{A^*_{11}} = [T_1]\mathbf{R}^{*H}_{A_{11}} = [T_1] = \begin{bmatrix}-(\sqrt{3}/12)a_1\\ (1/4)a_1\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}-(\sqrt{3}/12)a_1d_{11}+(1/4)a_1d_{12}+r_x\\ -(\sqrt{3}/12)a_1d_{21}+(1/4)a_1d_{22}+r_y\\ -(\sqrt{3}/12)a_1d_{31}+(1/4)a_1d_{32}+r_z\\ 1\end{bmatrix} \qquad (23)$$

$$r^*_{A_{12}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{A^*_{12}} = [T_1]\mathbf{R}_{A_{12}}^{*H}\,[T_1]\begin{bmatrix}-(\sqrt{3}/12)a_1\\ -(1/4)a_1\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}-(\sqrt{3}/12)a_1d_{11}-(1/4)a_1d_{12}+r_x\\ -(\sqrt{3}/12)a_1d_{21}-(1/4)a_1d_{22}+r_y\\ -(\sqrt{3}/12)a_1d_{31}-(1/4)a_1d_{32}+r_z\\ 1\end{bmatrix} \qquad (24)$$

$$r^*_{A_{13}} = \begin{bmatrix} x\\ y\\ z\\ 1\end{bmatrix}_{A^*_{13}} = [T_1]\mathbf{R}^{H}_{A_{23}} = [T_1]\begin{bmatrix}(\sqrt{3}/6)a_1\\ 0\\ 0\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}(\sqrt{3}/6)a_1d_{11}+r_x\\ (\sqrt{3}/6)a_1d_{21}+r_y\\ (\sqrt{3}/6)a_1d_{31}+r_z\\ 1\end{bmatrix} \qquad (25)$$

Note that,

$$\ell_i^2=(x_{A_{i2}}-x_{A_{o1}})^2+(y_{A_{i2}}-y_{A_{o1}})^2+(z_{A_{i2}}-z_{A_{o1}})^2,$$

for i=1.2..6, (26)

or

$$\ell_1^2=((\sqrt{3}/6)a_1d_{11}-(1/2)a_1d_{12}+r_x-(\sqrt{3}/6)a_o)^2$$
$$+((\sqrt{3}/6)a_1d_{21}-(1/2)a_1d_{22}+r_y+(a_o/2)^2$$
$$+((\sqrt{3}/6)a_1d_{31}-(1/2)a_1d_{32}+r_z) \qquad (27)$$

$$\ell_2^2= (x_{A^*_{11}}-x_{A_{o1}})^2+(y_{A^*_{11}}-y_{A_{o1}})^2+(z_{A^*_{11}}-z_{A_{o1}})^2 \qquad (28)$$

$$\ell_3^2= (x_{A_{13}}-x_{A_{o3}})^2+(y_{A_{13}}-y_{A_{o3}})^2+(z_{A_{13}}-z_{A_{o3}})^2 \qquad (29)$$

$$'\ell_4^2= (x_{A^*_{13}}-x_{A_{o3}})^2+(y_{A^*_{13}}-y_{A_{o3}})^2+(z_{A^*_{13}}-z_{A_{o3}})^2 \qquad (30)$$

$$\ell_5^2= (x_{A_{11}}-x_{A_{o2}})^2+(y_{A_{11}}-y_{A_{o2}})^2+(z_{A_{11}}-z_{A_{o2}})^2 \qquad (31)$$

$$\ell_6^2= (x_{A^*_{12}}-x_{A_{o2}})^2+(y_{A^*_{12}}-y_{A_{o2}})^2+(z_{A^*_{12}}-z_{A_{o2}})^2 \qquad (32)$$

$$\ell_2^2= (-(\sqrt{3}/12)a_1d_{11}+(1/4)a_1d_{12}+r_x-(\sqrt{3}/6)a_o)^2$$
$$+(-(\sqrt{3}/6)a_1d_{21}+(1/4)a_1d_{22}+r_y+(a_o/2))^2$$
$$+(-(\sqrt{3}/6)a_1d_{31}+(1/4)a_1d_{32}+r_z)^2 \qquad (33)$$

$$\ell_3^2=(-(\sqrt{3}/3)a_1d_{11}+r_x+(\sqrt{3}/3)a_0)^2+(-(\sqrt{3}/3)a_1d_{21}+r_y)^2+(-(\sqrt{3}/3)a_1d_{31}+r_z)^2 \quad (34)$$

$$\ell_4^2=((\sqrt{3}/6)a_1d_{11}+r_x+(\sqrt{3}/3)a_0)^2+((\sqrt{3}/6)a_1d_{21}+r_y)^2+((\sqrt{3}/6)a_1d_{31}+r_z)^2 \quad (35)$$

$$\ell_5^2=((\sqrt{3}/6)a_1d_{11}+(1/2)a_1d_{12}+r_x-(\sqrt{3}/6)a_0)^2+((\sqrt{3}/6)a_1d_{21}+(1/2)a_1d_{22}+r_y-(1/2)a_0)^2+((\sqrt{3}/6)a_1d_{31}+(1/2)a_1d_{32}+r_z)^2 \quad (36)$$

$$\ell_6^2=(-(\sqrt{3}/12)a_1d_{11}-(1/4)a_1d_{12}+r_x-(\sqrt{3}/6)a_0)^2+((\sqrt{3}/12)a_1d_{21}-(1/4)a_1d_{22}+r_y-(1/2)a_0)^2+(-(\sqrt{3}/12)a_1d_{31}-(1/4)a_1d_{32}+r_z)^2 \quad (37)$$

Equations (26)–(37) represent a set of equations for the solution of the inverse kinematics problem of such a robot manipulator.

Note that given a desired position of the origin of the T_1 frame in the upper said joint platform, i.e., r_x, r_y, r_z, and a desired orientation of it with respect to the base frame T_0, i.e.,θ, φ and ψ , the desired leg lengths ℓ_i, i=1,2,6 can be explicity determined. These ℓ_i's would then determine the extent of computer–controlled prismatic extension of the robot legs.

In our case the lengths ℓ_1, ℓ_3, and ℓ_5 are fixed and basiclly equal to some length ℓ_0. This means that equations (32), (34) and (36) now completely define the boundaries of the work space of the robot and equations (33), (35) and (37) can be used to determine the actuation lengths necessary to generate the desired attitude (position + orientation) of the upper platform. Furthermore, equations (32), (34), and (36) determine the values r_x, r_y and r_z as a function of θ, φ, and ψ given that ℓ_1, ℓ_3 and ℓ_5 are prescribed. Therefore, given the values of ℓ_1, ℓ_3 and ℓ_5 and the desired orientation of the frame T_1 with respect to T_0, equations (32)–(37) completely define an algorithm to achieve computer–controlled positioning of the first platform.

An exact similar analysis could be presented for the kinematics and the solution to the inverse kinematics problem of the second and if desired, the third platforms, respectively.

Extension To Multiple Platforms

Referring to Figure 3 below, we note that one may use a similar treatment for the frame T_2 or the said gripper platform with respect to the frames T_1 and T_0.

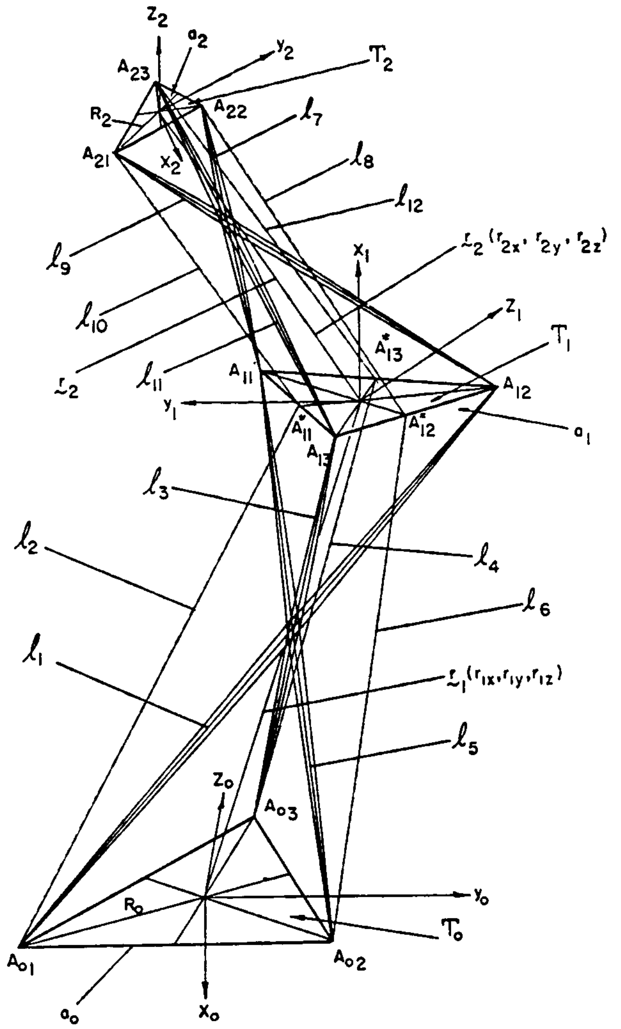


Figure 3– Kinematic structure of a 6–axis hybrid manipulator

Note that in this case

$$\ell_7^2= (x_{A_{22}}-x_{A_{11}})^2+(y_{A_{22}}-y_{A_{11}})^2+(z_{A_{22}}-z_{A_{11}})^2, \quad (38)$$

$$\ell_8^2= (x_{A_{22}}-x_{A^*_{12}})^2+(y_{A_{22}}-y_{A^*_{12}})^2+(z_{A_{22}}-z_{A^*_{12}})^2, \quad (39)$$

$$\ell_9^2=(x_{A_{21}}-x_{A_{12}})^2+(y_{A_{21}}-y_{12})^2+(z_{A_{21}}-z_{A^*_{12}})^2, \quad (40)$$

$$\ell_{10}^2=(x_{A_{21}}-x_{A^*_{11}})^2+(y_{A_{21}}-y_{A^*_{11}})^2+(z_{A_{21}}-z_{A^*_{11}})^2, \quad (41)$$

$$\ell_{11}^2=(x_{A_{23}}-x_{A_{13}})^2+(y_{A_{23}}-y_{A_{13}})^2+(z_{A_{23}}-z_{A_{13}})^2, \quad (42)$$

$$\ell_{12}^2=(x_{A_{23}}-x_{A^*_{13}})^2+(y_{A_{23}}-y_{A^*_{13}})^2+(z_{A_{23}}-z_{A^*_{13}})^2, \quad (43)$$

$$r_{A_{21}}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}_{A_{21}}= T_o^2\,R^H_{A_{21}}=T_o^2\begin{bmatrix}(\sqrt{3}/6)a_2\\-(a_2/2)\\0\\1\end{bmatrix}$$

$$=\begin{bmatrix}(\sqrt{3}/6)a_2d^*_{11}-(1/2)d^*_{12}+r_x\\(\sqrt{3}/6)a_2d^*_{21}-(1/2)a_2d^*_{32}+r_y\\(\sqrt{3}/6)a_2d_{31}-(1/2)a_2d_{22}+r_z\\ \end{bmatrix} \quad (44)$$

$$r_{A_{22}}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}_{A_{22}}=T_o^2\,R^H_{A_{22}}=T_o^2\begin{bmatrix}(\sqrt{3}/6)a_2\\(a_2/2)\\0\\1\end{bmatrix}$$

$$=\begin{bmatrix}(\sqrt{3}/6)a_2d^*_{11}+(1/2)a_2d^*_{12}+r_x\\(\sqrt{3}/6)a_2d^*_{21}+(1/2)a_2d^*_{32}+r_y\\(\sqrt{3}/6)a_2d_{31}+(1/2)a_2d_{22}+r_z\\ \end{bmatrix} \quad (45)$$

$$r_{A_{23}}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}_{A_{23}}= T_o^2\,R^H_{A_{23}}=T_o^2\begin{bmatrix}-(\sqrt{3}/3)a_2\\0\\0\\1\end{bmatrix}$$

$$=\begin{bmatrix}-(\sqrt{3}/3)a_2d^*_{11}+r_x\\-(\sqrt{3}/3)a_2d^*_{21}+r_y\\-(\sqrt{3}/3)a_2d_{31}+r_z\\1\end{bmatrix} \quad (46)$$

where d^*_{ij} are the direction cosines in T_o^2 transformations. Thus

$$\ell_7^2=((\sqrt{3}/6)a_2d^*_{11}-(1/2)a_2d^*_{12}+r_x-(\sqrt{3}/6)a_1d_{11}$$
$$-(1/2)a_1d_{12}-r_{1x})^2+((\sqrt{3}/6)a_2d^*_{21}-(1/2)a_2d_{32}$$
$$+r_y-(\sqrt{3}/6)a_1d_{21}-(1/2)a_1d_{22}-r_{1y})^2+((\sqrt{3}/6)a_2d^*_{31}$$
$$-(1/2)a_2d^*_{22}+r_z-(\sqrt{3}/6)a_1d_{31}-(1/2\_a_1d_{32}-r_{1z})^2 \quad (47)$$

$$\ell_8^2=((\sqrt{3}/6)a_2d^*_{11}+(1/2)a_2d^*_{12}+r_x+(\sqrt{3}/12)a_1d_{11}$$
$$+(1/4)a_1d_{12}-r_{1x})^2+((\sqrt{3}/6)a_2d^*_{21}+(1/2)a_2d^*_{32}$$
$$+r_y+(\sqrt{3}/12)a_1d_{21}+(1/4)a_1d_{22}-r_{1y})^2$$
$$+((\sqrt{3}/6)a_2d^*_{31}+(1/2)a_2d^*_{22}+r_z+(\sqrt{3}/12)a_1d_{31}$$
$$+(1/4)a_1d_{32}-r_{1z})^2 \quad (48)$$

$$\ell_9^2=((\sqrt{3}/6)a_2d^*_{11}-(1/2)a_2d^*_{12}+r_x-(\sqrt{3}/6)a_1d_{11}$$
$$+(1/2)a_1d_{12}-r_{1x})^2+((\sqrt{3}/6)a_2d^*_{21}$$
$$-(1/2)a_2d^*_{32}+r_y-(\sqrt{3}/6)a_1d_{21}$$
$$+(1/2)a_1d_{22}-r_{1y})^2+((\sqrt{3}/6)a_2d^*_{31}$$
$$-(1/2)a_2d^*_{22}+r_z-(\sqrt{3}/6)a_1d_{31}$$
$$+(1/2)a_1d_{32}-r_{1z})^2 \quad (49)$$

$$\ell_{10}^2=((\sqrt{3}/6)a_2d^*_{11}-(1/2)a_2d^*_{12}+r_x$$
$$+(\sqrt{3}/12)a_1d_{11}-(1/4)a_1d_{12}-r_{1x})^2$$
$$+((\sqrt{3}/6)a_2d^*_{21}-(1/2)a_2d^*_{32}+r_y+(\sqrt{3}/12)a_1d_{21}$$
$$-(1/4)a_1d_{22}-r_{1y})^2+((\sqrt{3}/6)a_1d^*_{31}$$
$$-(1/2)a_2d^*_{22}+r_z+(\sqrt{3}/12)a_1d_{31}$$
$$-(1/4)a_1d_{32}-r_{1z})^2 \quad (50)$$

$$\ell_{11}^2=((\sqrt{3}/3)a_2d^*_{11}+ r_x+ (\sqrt{3}/3)a_1d_{11}-r_{1x})^2$$
$$+((\sqrt{3}/3)a_2d^*_{21}+ r_y+ (\sqrt{3}/3)a_1d_{21}-r_y)^2$$
$$+((\sqrt{3}/3)a_2d^*_{31}+ \quad r_z+ \quad (\sqrt{3}/3)a_1d_{31}-r_{1z})^2 \quad (51)$$

$$\ell_{12}^2=((\sqrt{3}/3)a_2d^*_{11}+ r_x- (\sqrt{3}/6)a_1d_{11}-r_{1x})$$
$$+((\sqrt{3}/3)a_2d^*_{21}+ r_y- (\sqrt{3}/6)a_1d_{21}-r_{1y})^2$$
$$+((\sqrt{3}/3)a_2d^*_{31}+ r_z- (\sqrt{3}/6)a_1d_{31}-r_{1z})^2 \quad (52)$$

Note that the transformations T_o, T_1 and T_2 can also be expressed in terms of the associated Euler's angles θ, φ and ψ such that

$$d_{11}= C\theta,\ d_{12}= C\theta\, S\varphi\, S\psi - S\theta\, C\psi \quad (53)$$

$$d_{13}= C\theta\, S\varphi\, C\psi + S\varphi\, S\psi,\ d_{21}= S\varphi\, C\varphi \quad (54)$$

$$d_{22}= S\theta\, S\varphi\, S\psi + C\theta\, C\psi \quad (55)$$

$$d_{23} = S\theta\, S\varphi\, C\psi - C\varphi\, S\psi \qquad (56)$$

$$d_{31} = S\varphi,\; d_{32} = C\varphi\, S\psi,\; d_{33} = C\varphi\, C\psi. \qquad (57)$$

where the symbols C and S stand for Cosine and Sine of an angle.
Thus, all d_{ij}'s can be expressed in terms of three Euler's angle θ, φ and ψ [see chapter 2 of Shahinpoor [1]].

Now given the position and the orientation of the frame $\underset{\sim}{T}_2$ with respect to the reference base frame $\underset{\sim}{T}_o$ it is true that

$$\underset{\sim}{T}{}_o^2 = \underset{\sim}{T}{}_o^1\, \underset{\sim}{T}{}_1^2, \qquad (58)$$

where $\underset{\sim}{T}{}_j^i$ is the 4x4 homogeneous transformation describing the position and the orientations of frame $\underset{\sim}{T}_i$ with respect to frame $\underset{\sim}{T}_j$. In terms of the Euler's angles θ, φ, ψ, and θ_2, φ_2, ψ_2 and the position vectors $\mathbf{r}_1 = (r_{1x}, r_{1y}, r_{1z})$ and $\mathbf{r}_2 = (r_{2x}, r_{2y}, r_{2z})$, with respect to $\underset{\sim}{T}_o$ and $\underset{\sim}{T}_1$ frames, respectively, the following relationships hold true

$$\mathbf{Euler}\,(\theta_1, \varphi_1, \psi_1, r_{1x}, r_{1y}, r_{1z})\ \mathbf{Euler}\,(\theta_2, \varphi_2, \psi_2, r_{2x}, r_{2y}, r_{2z}) = \mathbf{Euler}\,(\theta, \varphi, \psi, r_x, r_y, r_z), \qquad (59)$$

where

$$\mathbf{T}_o^1 = \mathbf{Euler}\,(\theta_1, \varphi_1, \psi_1, r_{1x}, r_{1y}, r_{1z}) \qquad (60)$$

$$\mathbf{T}_1^2 = \mathbf{Euler}\,(\theta_2, \varphi_2, \psi_2, r_{2x}, r_{2y}, r_{2z}) \qquad (61)$$

$$\mathbf{T}_o^2 = \mathbf{Euler}\,(\theta, \varphi, \psi, r_x, r_y, r_z). \qquad (62)$$

Now given $\mathbf{T}_o^2$ in order to find the 6 actuation length ℓ_1, ℓ_3, ℓ_5, ℓ_8, ℓ_{10} and ℓ_{12} in terms of the known geometrical quantities ℓ_2, ℓ_4, ℓ_6, ℓ_7, ℓ_9, ℓ_{11}, a_o, a_1, a_2, one must solve 24 equations with 18 unknowns. The unknowns are θ_1, φ_1, ψ_1, r_{1x}, r_{1y}, r_{1z}, θ_2, φ_2, ψ_2, r_{2x}, r_{2y}, r_{2z}, ℓ_1, ℓ_3, ℓ_5, ℓ_8, ℓ_{10} and ℓ_{12}.
Note that under these circumstances

$$\mathbf{Euler}\,(\theta_1, \varphi_1, \psi_1, r_{1x}, r_{1y}, r_{1z}) = \begin{bmatrix} C\theta_1 & C\theta_1 S\varphi_1 S\psi_1 - S\theta_1 C\psi_1 & C\theta_1 S\varphi_1 C\psi_1 + S\varphi_1 S\psi_1 & r_{1x} \\ S\varphi_1 C\varphi_1 & S\theta_1 S\varphi_1 S\psi_1 + C\theta_1 C\psi_1 & S\theta_1 S\varphi_1 C\psi_1 - C\varphi_1 S\psi_1 & r_{1y} \\ -S\varphi_1 & C\varphi_1 S\psi_1 & C\varphi_1 C\psi_1 & r_{1z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (63)$$

$$\mathbf{Euler}\,(\theta_2, \varphi_2, \psi_2, r_{2x}, r_{2y}, r_{2z}) = \begin{bmatrix} C\theta_2 & C\theta_2 S\varphi_2 S\psi_2 - S\theta_2 C\psi_2 & C\theta_2 S\varphi_2 C\psi_2 + S\varphi_2 S\psi_2 & r_{2x} \\ S\varphi_2 C\varphi_2 & S\theta_2 S\varphi_2 S\psi_2 + C\theta_2 C\psi_2 & S\theta_2 S\varphi_2 C\psi_2 - C\varphi_2 S\psi_2 & r_{2y} \\ -S\varphi_2 & C\varphi_2 S\psi_2 & C\varphi_2 C\psi_2 & r_{2z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (64)$$

$$\mathbf{Euler}\,(\theta, \varphi, \psi, r_x, r_y, r_z) = \begin{bmatrix} C\theta & C\theta S\phi S\Psi - S\theta C\Psi & C\theta S\phi C\Psi + S\phi S\Psi & r_x \\ S\phi C\phi & S\theta S\phi S\Psi + C\theta C\Psi & S\theta S\phi C\Psi - C\phi S\Psi & r_y \\ -S\phi & C\phi S\Psi & C\phi C\Psi & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (65)$$

Thus

11→ $$C\theta = C\theta_1 C\theta_2 + S\varphi_2 C\varphi_2 (C\varphi_1 S\varphi_1 S\psi_1 - S\theta_1 C\psi_1) - S\varphi(C\theta_1 S\varphi_1 C\psi_1 + S\varphi_1 S\psi_1) \qquad (66)$$

12→ $$C\theta S\varphi S\psi - S\theta C\psi = C\theta_1(C\theta_2 S\varphi_2 S\psi_2 - S\theta_2 C\psi_2) + (C\theta_1 S\varphi_1 S\psi_1 - S\theta_1 C\psi_1)(S\theta_2 S\theta_2 S\psi_2 + C\theta_2 C\psi_2) + C\varphi_2 S\psi_2(C\theta_1 S\varphi_1 C\psi_1 + S\varphi_1 S\psi_1) \qquad (67)$$

13→ $$C\theta S\varphi C\psi + S\varphi S\psi = C\theta_1(C\theta_2 S\varphi_2 C\psi_2 + S\varphi_2 S\psi_2) + (C\theta_1 S\varphi_1 S\psi_1 - S\theta_1 C\psi_1)(S\theta_2 S\varphi_2 C\psi_2 - C\varphi_2 S\psi_2) + C\varphi_2 C\psi_2(C\theta_1 S\theta_1 C\psi_1 + S\varphi_1 S\psi_1) \qquad (68)$$

14→ $$r_x = r_{2x}\, C\theta_1 + r_{2y}(C\theta_1 S\varphi_1 S\psi_1 - S\theta_1 C\psi_1) + r_{2z}(C\theta_1 S\varphi_1 C\psi_1 + S\varphi_1 S\psi_1) + r_{1x} \qquad (69)$$

24→ $$r_y = r_{2x}(S\varphi_1 C\varphi_1) + r_{2y}(S\theta_1 S\varphi_1 S\psi_1 + C\theta_1 C\psi_1)\; r_{2z}(S\theta_1 S\varphi_1 C\psi_1 - C\varphi_1 S\psi_1) + r_{1y} \qquad (70)$$

34→ $$r_z = r_{2x}(-S\varphi_1) + r_{2y}(C\varphi_1 S\psi_1) + r_{2z}(C\varphi_1 C\psi_1) + r_{1z} \qquad (71)$$

21→ $$S\varphi C\varphi = C\theta_2(S\varphi_1 C\varphi_1) + S\varphi_2 C\varphi_2(S\theta_1 S\varphi_1 S\psi_1 + C\theta_1 C\psi_1) - S\varphi(S\theta_1 S\varphi_1 C\psi_1 - C\theta_1 S\psi_1) \qquad (72)$$

31→ $$S\varphi = C\theta_2(-S\varphi_1)+S\varphi_2C\varphi_2(C\varphi_1S\psi_1)-S\varphi_2C\varphi_1C\psi_1 \tag{73}$$

32→ $$\begin{aligned}C\varphi S\psi = &-S\varphi_1(C\theta_2S\varphi_2S\psi_2 - S\theta_2C\psi_2)\\ &+C\varphi_1S\psi_1(S\theta_2S\varphi_2S\psi_2 + C\theta_2C\psi_2)\\ &+C\psi_1C\psi_1\ C\varphi_2S\psi_2\end{aligned} \tag{74}$$

33→ $$\begin{aligned}C\varphi C\psi = &-S\varphi_1(C\theta_2S\varphi_2C\psi_2 + S\varphi_2S\psi_2)\\ &+C\varphi_1S\psi_1(S\theta_2S\varphi_2C\psi_2 - C\varphi_2S\psi_2)\\ &+C\varphi_1C\psi_1C\varphi_2C\psi_2\end{aligned} \tag{75}$$

22→ $$\begin{aligned}&S\theta S\varphi S\psi + C\theta C\psi = S\varphi_1C\varphi_1(C\theta_2S\varphi_2S\psi_2 -\\ &S\theta_2C\psi_2)+(S\theta_1S\varphi_1S\psi_1 + C\theta_1C\psi_1)(S\theta_2S\varphi_2S\psi_2\\ &+C\theta_2C\psi_2)C\varphi_2S\psi_2(S\theta_1S\varphi_1C\psi_1-C\varphi_1S\psi_1)\end{aligned} \tag{76}$$

23→ $$\begin{aligned}&S\theta S\varphi C\psi - C\varphi S\psi = S\varphi_1C\varphi_1(C\theta_2S\varphi_2C\psi_2 +\\ &S\varphi_2S\psi_2)+(S\theta_1S\varphi_1S\psi_1 + C\theta_1C\psi_1)(S\theta_2S\varphi_2C\psi_2 -\\ &C\varphi_2S\psi_2)+C\varphi_2C\psi_2(S\theta_1S\varphi_1C\psi_1 - C\varphi_1S\psi_1).\end{aligned} \tag{77}$$

In addition to the above equations the following equations are also true:

$$\begin{aligned}\ell_1^2 =&[(\sqrt{3}/6)a_1C\theta_1-(1/2)a_1(C\theta_1S\varphi_1S\psi_1-S\theta_1C\psi_1)\\ &+r_{1x}- (\sqrt{3}/6)a_o)^2+[(\sqrt{3}/6)a_1S\varphi_1C\varphi_1\\ &-(1/2)a_1(S\theta_1S\varphi_1S\psi_1+C\theta_1C\psi_1)+r_{1y}+(a_o/2)]^2\\ &+[-(\sqrt{3}/6)a_1S\varphi_1-(1/2)a_1C\varphi_1S\psi_1+r_{1z}]]^2\end{aligned} \tag{78}$$

$$\begin{aligned}\ell_2^2=&[-(\sqrt{3}/12)a_1C\theta_1+(1/4)a_1(C\theta_1S\varphi_1S\psi_1\\ &-S\theta_1C\psi_1)+r_{1x}- (\sqrt{3}/6)a_o]^2+[-(\sqrt{3}/12)a_1S\varphi_1\\ &+(1/4)a_1(S\theta_1S\varphi_1S\psi_1+C\theta_1C\psi_1)+r_{1y}+(a_o/2]^2\\ &+[+ (\sqrt{3}/12)a_1S\varphi_1+(1/4)a_1C\varphi_1S\psi_1+r_{1z}]^2\end{aligned} \tag{79}$$

$$\begin{aligned}\ell_3^2 =&[- (\sqrt{3}/3)a_1C\theta_1+r_{1x}+ (\sqrt{3}/3)a_o]^2+[-(\sqrt{3}/3)\\ &a_1S\varphi_1C\varphi_1+r_{1y}]^2+[+(\sqrt{3}/3)a_1S\varphi_1+r_{1z}]^2\end{aligned} \tag{80}$$

$$\begin{aligned}\ell_4^2 =&[(\sqrt{3}/6)a_1C\theta_1+r_{1x}+ (\sqrt{3}/3)a_o]^2\\ &+[(\sqrt{3}/6)a_1S\varphi_1C\varphi_1+r_{1y}]^2\\ &+[- (\sqrt{3}/6)a_1S\varphi_1+r_{1z}]^2\end{aligned} \tag{81}$$

$$\begin{aligned}\ell_5^2=&[(\sqrt{3}/6)a_1C\theta_1+(1/2)a_1(C\theta_1S\varphi_1S\psi_1\\ &-S\theta_1C\psi_1)+r_{1x}- (\sqrt{3}/6)a_o]^2+[+(\sqrt{3}/6)a_1S\varphi_1C\varphi_1\\ &+(1/2)a_1(S\theta_1S\varphi_1S\psi_1+C\theta_1C\psi_1)+r_{1y}-(1/2)a_o]^2\\ &+[- (\sqrt{3}/6)a_1S\varphi_1+(1/2)a_1C\varphi_1S\psi_1+r_{1z}]^2\end{aligned} \tag{82}$$

$$\begin{aligned}\ell_6^2=&[-(\sqrt{3}/12)a_1C\theta_1-(1/4)a_1(C\theta_1S\varphi_1S\psi_1\\ &-S\theta_1C\psi_1)+r_{1x}-(\sqrt{3}/6)a_o]^2+[\\ &-(\sqrt{13}/12)a_1S\varphi_1C\varphi_1-(1/4)a_1(S\theta_1S\varphi_1S\psi_1\\ &+C\theta_1C\psi_1)+r_{1y}-(1/2)a_o]^2+[\sqrt{3}/12)a_1S\varphi_1\\ &-(1/4)a_1C\varphi_1S\varphi_1+r_{1z}]^2\\ &-(1/2)a_2C\varphi_2S\psi_2+r_y-(\sqrt{3}/6)a_1S\varphi_1C\varphi_1\\ &-(1/2)a_1(S\theta_1S\varphi_1S\psi_1+C\theta_1C\psi\ )-r_{1y}]^2\end{aligned} \tag{83}$$

$$\begin{aligned}\ell_7^2=&[(\sqrt{3}/6)a_2C\theta_2-(1/2)a_2(C\theta_2S\varphi_2S\psi_2-S\theta_2C\psi_2)\\ &+r_x-(\sqrt{3}/6)a_1C\theta_1-(1/2)a_1(C\theta_1S\varphi_1S\psi_1\\ &-S\theta_1C\psi_1)-r_{1x}]^2+[(\sqrt{3}/6)a_2S\varphi_2C\varphi_2\\ &-(1/2)a_2C\varphi_2S\psi_2+r_y-(\sqrt{3}/6)a_1S\varphi_1C\varphi_1\\ &-(1/2)a_1(S\theta_1S\varphi_1S\psi_1+C\theta_1C\psi_1)-R_{1y}]^2\\ &+[-(\sqrt{3}/6)a_2S\varphi_2-(1/2)a_2(S\theta_2S\varphi_2S\psi_2\\ &+C\theta_2C\psi_2)+r_z+(\sqrt{3}/6)a_1S\varphi_1\\ &-(1/2)a_1C(_1S\psi_1-r_{1z}]^2\end{aligned} \tag{84}$$

Similar expressions follow for $\ell_8{}^2$ through ℓ_{12}^2.

REFERENCES

1– M. Shahinpoor, "A Robot Engineering Textbook", Harper & Row Publlishers, New York, London, (1987)

2– E.F. Fichter, "Kinematics of a Parallel–Connection Manipulator", ASME paper 84–DET–45, (1984).

3– E.F. Fichter, "A Stewart Platform–Based Manipulator General Theory and Practical Construction", Int. J. Robotics. Res., vol.5 no. 2, pp 165–190, (1986).

4– D. Stewart, "A Platform With Six Degrees of Freedom", Proc. Inst. Mech. Eng., vol. 180, no. 1, pp. 371–386, (1965).

5– C.F. Earl and J. Rooney, "Some Kinematic Structures For Robot Manipulator Designs", Trans. ASME, J. Mechanism, Transmissions and Automation in Design, vol. 105, pp. 15–22, (1983).

6– K.H. Hunt, "Structural Kinematics of In–Parallel–Acturated Robot–Arms", Tran. ASME, J. Mechanisms, Transmissions and Automation in Designs, vol. 105, pp. 705–712, (1983).

7– D.C.H. Yan and T.W. Lee, "Feasibility Study of A Platform Type of Robotic Manipulator From A Kinematic Viewpoint", Trans. ASME, J.Mechanisms, Transimissions and Automation In Design," vol. 106, pp. 191–198, (1984).

N91-20645

FORWARD AND INVERSE KINEMATICS OF DOUBLE UNIVERSAL JOINT ROBOT WRISTS

Dr. Robert L. Williams II
Automation Technology Branch, M.S. 152D
NASA Langley Research Center
Hampton, VA 23665-5225

ABSTRACT

A robot wrist consisting of two universal joints can eliminate the wrist singularity problem found on many industrial robots. This paper presents forward and inverse position and velocity kinematics for such a wrist having three degrees of freedom. Denavit-Hartenberg parameters are derived to find the transforms required for the kinematic equations. The Omni-Wrist, * a commercial double universal joint robot wrist, is studied in detail. There are four levels of kinematic parameters identified for this wrist; three forward and three inverse maps are presented for both position and velocity. These equations relate the hand coordinate frame to the wrist base frame. They are sufficient for control of the wrist standing alone.

When the wrist is attached to a manipulator arm, the offset between the two universal joints complicates the solution of the overall kinematics problem. All wrist coordinate frame origins are not coincident, which prevents decoupling of position and orientation for manipulator inverse kinematics. This is a topic for future research.

INTRODUCTION

Many current industrial robot wrists suffer from singularity limitations where at least two wrist coordinate frames align, reducing orientational freedom. Near singular positions, extremely large joint rates are required to maintain constant cartesian rates. One proposed wrist design for reducing singularities uses the universal joint to achieve roll, pitch, and yaw orientation. An overview of robot wrists, including universal joint designs, is given by Rosheim (1989). Other references present non-singular robot wrist designs, e.g., (Barker, 1986), (Milenkovic, 1987), (Rosheim, 1987 and 1986), and (Trevelyan, 1986).

McKinney (1988) presents forward kinematic and resolved rate equations for single and double universal joint robot wrists. The author studies a specific double universal joint wrist, the Omni-Wrist from Ross-Hime Designs, Inc. A single universal joint wrist is attractive because its motion is purely rotational. However, the workspace is limited due to gimbal lock singularities. Also, the roll velocity of the output shaft is variable, given a constant input roll rate. Therefore, a wrist with two universal joints in series is suggested, which allows an approximately hemispherical singularity-free workspace (McKinney, 1988). Two universal joints yield a constant roll velocity ratio (Mabie and Reinholtz, 1987).

The current paper presents forward and inverse kinematic position and velocity equations for control of double universal joint robot wrists. Denavit-Hartenberg parameters are presented for double universal joint wrists. The Omni-Wrist kinematic transformations are presented. Four levels of kinematic parameters are identified, from the actuator angles to the position and orientation of the hand. Three mappings are presented for each of the forward position, inverse position, forward velocity, and inverse velocity (resolved rate) problems. These equations relate the robot hand to the robot wrist base and are sufficient for control of the wrist standing alone.

The double universal joint wrist is not purely rotational due to the offset between the two universal joints. Position and orientation trajectories thus may not be decoupled for a double universal joint wrist attached to a manipulator arm. The manipulator inverse position and velocity problems are more complicated for the double universal joint robot wrist than a purely rotational robot wrist.

SYMBOLS

$\{m\}$	Cartesian coordinate frame m
$\{3\}$	Wrist base coordinate frame
$\{8\}$	Hand coordinate frame
$\theta_{4A}, \theta_{5A}, \theta_{6A}$	Actuator angles
$\theta_{4G}, \theta_{5G}, \theta_{6G}$	Gear bail angles
$\theta_4, \theta_5, \theta_6$	Universal joint angles
$[^n_mT]$	Homogeneous transformation matrix of $\{m\}$ relative to $\{n\}$
$[^n_mR]$	Rotation matrix of $\{m\}$ relative to $\{n\}$
r_{ij}	Element (i,j) of $[^3_8R]$
$\{^nP_m\}$	Position vector from origin of $\{n\}$ to $\{m\}$, expressed in $\{n\}$
$\{^n\hat{X}_m\}$	Unit direction vector X of $\{m\}$ expressed in $\{n\}$
$\{^m\omega_m\}$	Angular velocity of $\{m\}$ with respect to $\{3\}$, expressed in $\{m\}$
$\{^mv_m\}$	Linear velocity of $\{m\}$ origin with respect to $\{3\}$, expressed in $\{m\}$
$F1$	Forward map solving θ_{iG} given θ_{iA}, i=4,5,6
$F2$	Forward map solving θ_i given θ_{iG}, i=4,5,6
$F3$	Forward map solving $[^3_8T]$ given θ_i, i=4,5,6
$I1, I2, I3$	Inverses of $F3, F2, F1$, respectively
$FVi, IVi, i = 1,2,3$	Forward and Inverse velocity maps, defined analogously
$\dot{\theta}_i$	Joint rate i
c_i	$cos\theta_i$
s_i	$sin\theta_i$
t_i	$tan\theta_i$
L	Offset length between the universal joints

DOUBLE UNIVERSAL JOINT WRIST KINEMATICS

A universal joint is used to transfer rotations between intersecting shafts. Most kinematics textbooks discuss universal joints (e.g. Mabie and Reinholtz, 1987). A kinematic diagram for the double universal joint robot wrist is shown in Fig. 1. The input shaft rotates about a fixed axis and the output shaft is free; thus there are five degrees of freedom. Coupling of θ_5 and θ_6 reduces this number to three degrees of freedom.

* The mention herein of a trademark of a commercial product does not constitute any recommendation for use by the Government.

Figure 1 shows the initial position for all wrist coordinate frames; all universal joint angles are zero in this configuration. Frame $\{3\}$ is the wrist base frame, fixed for this paper. Frame $\{4\}$ rotates by θ_4 relative to $\{3\}$; $\{5\}$ rotates by θ_5 relative to $\{4\}$; $\{6\}$ rotates by θ_6 relative to $\{5\}$; $\{7\}$ rotates by the coupled θ_6 relative to $\{6\}$; and the hand frame $\{8\}$ rotates by the coupled θ_5 relative to $\{7\}$.

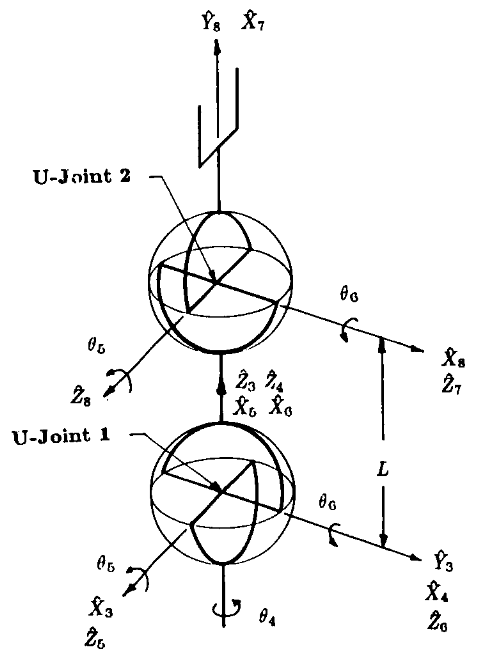


Figure 1
Double Universal Joint Robot Wrist Kinematic Diagram

Denavit-Hartenberg Parameters

The Denavit-Hartenberg parameters for the double universal joint robot wrist of Fig. 1 are given in Table I, which follows the convention in Craig (1988).

Table I Denavit-Hartenberg Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
4	0	0	0	$\theta_4 + 90°$
5	90°	0	0	$\theta_5 + 90°$
6	90°	0	0	θ_6
7	0	L	0	θ_6
8	-90°	0	0	$\theta_5 - 90°$

Forward Position

The forward solution finds $[{}^{3}_{8}T]$ given θ_4, θ_5, θ_6. Equation 1 is the homogeneous transformation matrix describing the position and orientation of $\{i\}$ with respect to $\{i-1\}$ (Craig, 1988).

$$[{}^{i-1}_{\ \ i}T] = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1)$$

Five homogeneous transformation matrices relating $\{3\}$ through $\{8\}$ are obtained by substituting the Denavit-Hartenberg parameters into Eq. 1.

$$[{}^{3}_{4}T] = \begin{bmatrix} -s_4 & -c_4 & 0 & 0 \\ c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [{}^{4}_{5}T] = \begin{bmatrix} -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[{}^{5}_{6}T] = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [{}^{6}_{7}T] = \begin{bmatrix} c_6 & -s_6 & 0 & L \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[{}^{7}_{8}T] = \begin{bmatrix} s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The general forward kinematics solution is Eq. 2. The (4×4) forward transform is comprised of a (3 × 3) rotation matrix representing the orientation and a (3 × 1) position vector locating the origin of $\{8\}$ in $\{3\}$. The specific terms are given in Eq. 3.

$$[{}^{3}_{8}T] = \left[\begin{array}{ccc|c} & [{}^{3}_{8}R] & & \{{}^{3}P_8\} \\ --- & --- & --- & --- \\ 0 & 0 & 0 & 1 \end{array}\right] \qquad (2)$$

$$[{}^{3}_{8}T] = [{}^{3}_{4}T(\theta_4)]\,[{}^{4}_{5}T(\theta_5)]\,[{}^{5}_{6}T(\theta_6)]\,[{}^{6}_{7}T(\theta_6)]\,[{}^{7}_{8}T(\theta_5)]$$

$$[{}^{3}_{8}T] = \begin{bmatrix} 2s_5c_6K1 - s_4 & 2c_5c_6K1 & -2s_6K1 + c_4 & L(K1) \\ 2s_5c_6K2 + c_4 & 2c_5c_6K2 & -2s_6K2 + s_4 & L(K2) \\ 2s_5c_5c_6^2 & 2c_5^2c_6^2 - 1 & -2c_5s_6c_6 & Lc_5c_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

$K1 = c_4s_6 + s_4s_5c_6$

$K2 = s_4s_6 - c_4s_5c_6$

Inverse Position

The inverse problem solves for the universal joint angles given task space input. The full $[{}^{3}_{8}T]$ cannot be specified because it has six freedoms, and the wrist only three. Due to the following constraint, which dictates that $\{{}^{3}P_8\}$ travel on the surface of a sphere of radius L, $\{{}^{3}P_8\}$ cannot be the input, because it has two independent freedoms.

$$P_x^2 + P_y^2 + P_z^2 = L^2 \qquad (4)$$

The rotation matrix $[{}^{3}_{8}R]$ is the input to the inverse problem.

$$[{}^{3}_{8}R] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (5)$$

$$[{}^{3}_{8}R] = [{}^{3}_{4}R(\theta_4)]\,[{}^{4}_{5}R(\theta_5)]\,[{}^{5}_{6}R(\theta_6)]\,[{}^{6}_{7}R(\theta_6)]\,[{}^{7}_{8}R(\theta_5)]$$

The angle θ_4 is isolated by inverting $[{}^{3}_{4}R]$ and multiplying it on the left of both sides of Eq. 5.

$$[{}^{3}_{4}R(\theta_4)^{-1}]\,[{}^{3}_{8}R] = [{}^{4}_{5}R(\theta_5)]\,[{}^{5}_{6}R(\theta_6)]\,[{}^{6}_{7}R(\theta_6)]\,[{}^{7}_{8}R(\theta_5)] \qquad (6)$$

The angles θ_5 and θ_6 are eliminated from the right hand side of Eq. 6 by equating the (1,1), (2,3), and (3,2) elements, given in Eqs. 7, 8, and 9, respectively.

$$r_{21}c_4 - r_{11}s_4 = -2s_5^2c_6^2 + 1 \tag{7}$$
$$-r_{13}c_4 - r_{23}s_4 = -2c_6^2 + 1 \tag{8}$$
$$r_{32} = 2c_5^2c_6^2 - 1 \tag{9}$$

Equation 7 is subtracted from Eq. 9 to eliminate θ_5.

$$r_{32} - r_{21}c_4 + r_{11}s_4 = 2c_6^2 - 2 \tag{10}$$

Equations 8 and 10 are added to remove θ_6.

$$\begin{aligned} E cos\theta_4 + F sin\theta_4 + G = 0 \\ E = -(r_{13} + r_{21}) \\ F = r_{11} - r_{23} \\ G = r_{32} + 1 \end{aligned} \tag{11}$$

Using the tan half angle substitution (Mabie and Reinholtz, 1987), there are two solutions for θ_4:

$$\theta_{4_{1,2}} = 2tan^{-1}\left[\frac{-F \pm \sqrt{E^2+F^2-G^2}}{G-E}\right] \tag{12}$$

The radicand in Eq. 12 is simplified with orthonormal constraints. The columns of $[^3_8R]$ are the X, Y, Z unit vectors of $\{8\}$ expressed in $\{3\}$ coordinates, while the rows are those of $\{3\}$ given in $\{8\}$. The orthogonal constraints dictate that both the columns and rows of a rotation matrix form a dextral mutually perpendicular set. The normal constraints dictate that the length of all columns and rows is unity. With the following four constraints, $E^2 + F^2 - G^2 = 0$.

$$\begin{aligned} {}^3\hat{Y}_8 = {}^3\hat{Z}_8 \times {}^3\hat{X}_8 \\ \left|{}^8\hat{X}_3\right| = 1.0 \\ \left|{}^3\hat{Y}_8\right| = 1.0 \\ \left|{}^8\hat{Y}_3\right| = 1.0 \end{aligned} \tag{13}$$

Therefore, given $[^3_8R]$, there is one solution to θ_4 (two repeated roots), from Eq. 12.

$$\theta_4 = 2tan^{-1}\left[\frac{-F}{G-E}\right] \tag{14}$$

With θ_4 solved, the left hand side of Eq. 6 is known. The next step is to isolate and solve θ_5.

$$[^4_5R(\theta_5)^{-1}]\,[^3_4R(\theta_4)^{-1}]\,[^3_8R]\,[^7_8R(\theta_5)^{-1}] = [^5_6R(\theta_6)]\,[^6_7R(\theta_6)] \tag{15}$$

The (2,2) elements of Eq. 15 are equated to solve θ_5.

$$\theta_5 = tan^{-1}\left[\frac{r_{13}s_4 - r_{23}c_4}{r_{33}}\right] \tag{16}$$

Both solutions from the inverse tangent function are mathematically valid, due to symmetry: θ_5 $(\frac{-\pi}{2} < \theta_5 < \frac{-\pi}{2})$ and $\theta_5 + \pi$. When θ_4 and θ_5 are known the left hand side of Eq. 15 is known. Angle θ_6 is solved by equating the (3,2) elements of Eq. 15.

$$2\theta_6 = cos^{-1}(r_{13}c_4 + r_{23}s_4) \tag{17}$$

The inverse cosine function solution is $\pm 2\theta_6$. This ambiguity is resolved by determining which sign satisfies the (1,2) terms of Eq. 15.

$$s2\theta_6 = (r_{23}c_4 - r_{13}s_4)s_5 - r_{33}c_5 \tag{18}$$

The proper sign for $2\theta_6$ is chosen from Eq. 18. Another valid solution for $2\theta_6$ is $2\theta_6 + 2\pi$; therefore, a second mathematical solution for θ_6 is $\theta_6 + \pi$.

A generalization is drawn regarding the two solutions for θ_5. The right hand side of Eq. 18 for $\theta_5 + \pi$ is the negative of that value for θ_5. Therefore, the value of θ_6 corresponding to $\theta_5 + \pi$ is the negative of θ_6 corresponding to θ_5.

There are four solutions to the inverse problem: a unique θ_4, two θ_5 for this θ_4, plus two θ_6 for each θ_5. Only one combination need be solved; the remaining three are formed from the structure of the solution, summarized in Table II. In rows 1 and 2 of Table II, θ_6 can be positive or negative; the negative θ_6 in rows 3 and 4 indicates opposite sign to θ_6 of row 1 .

Table II Inverse Position Solutions

Solution	θ_4	θ_5	θ_6
1	θ_4	θ_5	θ_6
2	θ_4	θ_5	$\theta_6 + \pi$
3	θ_4	$\theta_5 + \pi$	$-\theta_6$
4	θ_4	$\theta_5 + \pi$	$-\theta_6 + \pi$

Forward Velocity

The forward velocity problem solves for cartesian rates given joint rates using velocity recursion equations (Craig, 1988).

$$\{^{i+1}\omega_{i+1}\} = [^{i+1}_iR]\{^i\omega_i\} + \dot{\theta}_{i+1}\{^{i+1}\hat{Z}_{i+1}\} \tag{19}$$
$$\{^{i+1}v_{i+1}\} = [^{i+1}_iR]\left(\{^iv_i\} + \{^i\omega_i\} \times \{^iP_{i+1}\}\right) \tag{20}$$

The wrist Jacobian matrix is extracted from the forward velocity solution. The (6×3) Jacobian matrix maps the (3×1) joint rates into the (6×1) cartesian rates. The Jacobian matrix is partitioned into (3×3) rotational and translational Jacobian matrices, $[J_R]$ and $[J_T]$.

$$\begin{aligned} \{^8\omega_8\} &= [J_R]\{\dot{\theta}\} \\ [J_R] &= \begin{bmatrix} s2\theta_5c_6^2 & s_5s2\theta_6 & 2c_5 \\ 2c_5^2c_6^2 - 1 & c_5s2\theta_6 & -2s_5 \\ -c_5s2\theta_6 & 2c_6^2 & 0 \end{bmatrix} \\ \{\dot{\theta}\} &= \begin{Bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{Bmatrix} \end{aligned} \tag{21}$$

$$\begin{aligned} \{^8v_8\} &= [J_T]\{\dot{\theta}\} \\ [J_T] &= L\begin{bmatrix} s_6 & -c_5c_6 & s_5s_6 \\ 0 & s_5c_6 & c_5s_6 \\ s_5c_6 & 0 & c_6 \end{bmatrix} \end{aligned} \tag{22}$$

Inverse Velocity (Resolved Rate)

The resolved rate problem solves for joint rates given cartesian rates. This problem is overconstrained (six equations in three unknowns). Therefore, only three cartesian rates may be specified. The resolved rate input cannot be $\{^8v_8\}$ because $[J_T]$ is always singular. This is due to the constraint, Eq. 4.

$$|J_T| = Ls_5s_6c_6^2(1 - c_5^2 - s_5^2) = 0 \tag{23}$$

The inverse velocity problem is solved by inverting Eq. 21.

$$\{\dot{\theta}\} = [J_R^{-1}]\{{}^8\omega_8\}$$

$$[J_R^{-1}] = \begin{bmatrix} t_5 & 1 & \frac{-t_6}{c_5} \\ s_5 t_6 & c_5 t_6 & 1 - \frac{1}{2c_6^2} \\ c_5 - \frac{1}{2c_5} & -s_5 & \frac{1}{2} t_5 t_6 \end{bmatrix} \tag{24}$$

The wrist singularity conditions are found by setting the determinant to zero.

$$|J_R| = 4c_5c_6^2 = 0 \tag{25}$$

The double universal joint robot wrist is singular when $\theta_5 = \pm\frac{\pi}{2}$ or $\theta_6 = \pm\frac{\pi}{2}$.

OMNI-WRIST KINEMATICS

The Omni-Wrist by Ross-Hime Designs, Inc. is a double universal joint robot wrist. Figure 2 displays a section view of the Omni-Wrist. Planetary gears transfer the first universal joint rotations θ_5 and θ_6 to the second universal joint.

The rotational axes for θ_5 and θ_6 are moving. The Omni-Wrist has outer and inner bevel gear bails to transfer rotations from two actuators fixed in the wrist base to the angles θ_5 and θ_6, to avoid moving actuators. No intermediate gear bail is required for θ_4 because it rotates about an axis fixed in the wrist base. In addition to the outer and inner gear bails, helical gear trains are used to reduce the speed and amplify the torque for each of the three actuators.

Referring to Fig. 2, actuator 1 drives θ_4. The inner gear bail rotates in the plane of the paper; the outer rotates about a perpendicular axis. The inner gear bail angle, rotated by actuator 2, equals θ_6 when $\theta_4 = \theta_5 = 0$. Actuator 3 rotates the outer gear bail, whose angle equals θ_5 when $\theta_4 = \theta_6 = 0$. In general, the inner and outer gear bails combine to yield θ_5 and θ_6.

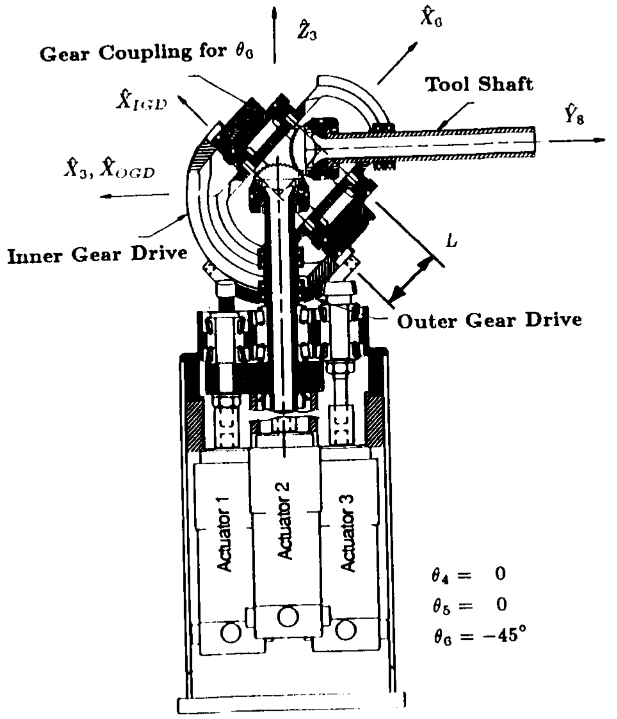


Figure 2
Omni-Wrist Section View

The roll angle θ_4 is continuous and bidirectional. The inner and outer gear bail angles are limited to $\pm 45°$. These limits apply to θ_5 when $\theta_4 = \theta_6 = 0$, and to θ_6 when $\theta_4 = \theta_5 = 0$. When these angles are not zero, the limits on θ_5 and θ_6 are more restrictive.

There are four levels of Omni-Wrist kinematic parameters: 1) Actuator angles $(\theta_{4A}, \theta_{5A}, \theta_{6A})$; 2) Gear bail angles $(\theta_{4G}, \theta_{5G}, \theta_{6G})$; 3) Universal joint angles $(\theta_4, \theta_5, \theta_6)$; and 4) Hand coordinate frame $[{}^3_8T]$. All angles are zero in the initial position.

Omni-Wrist Position Kinematics

Figure 3 describes the three forward and inverse position mappings between the four levels of Omni-Wrist kinematic parameters. The overall forward position problem finds $[{}^3_8T]$ given the actuator angles, using maps $F1, F2$, and $F3$. The inverse position problem finds the actuator angles given $[{}^3_8R]$ via the maps $I1, I2$, and $I3$.

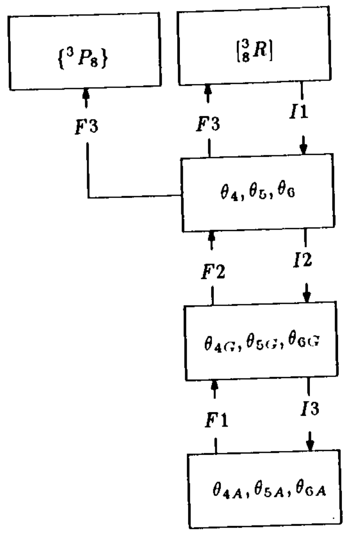

$\{{}^3P_8\}$
$[{}^3_8R]$
F3
F3
I1
$\theta_4, \theta_5, \theta_6$
F2
I2
$\theta_{4G}, \theta_{5G}, \theta_{6G}$
F1
I3
$\theta_{4A}, \theta_{5A}, \theta_{6A}$

Figure 3
Position Mappings

Maps $F3$ and $I1$ are the general wrist solutions, Eq. 3 and Eqs. 14, 16, and 17, respectively. The remaining maps are developed in this section.

Position Maps $F1$ and $I3$

The gear bail angles are related to the actuator angles by gear trains. Forward map $F1$ is given in Eq. 26.

$$\theta_{4G} = N_1\theta_{4A} \tag{26a}$$
$$\theta_{5G} = N_2\theta_{5A} \tag{26b}$$
$$\theta_{6G} = N_3\theta_{6A} \tag{26c}$$

For the Omni-Wrist, $N_1 = \frac{-1}{175.5}$, $N_2 = \frac{1}{259.2}$, and $N_3 = \frac{-1}{220.0}$. The map $I3$ is the inverse of Eqs. 26.

Position Maps $F2$ and $I2$

The kinematic relationships between the gear bail angles and the universal joint angles are coupled and transcendental. McKinney (1988) solved a problem equivalent to $I2$; the map $F2$ was not solved.

Two coordinate frames are introduced to determine the kinematic relationships between the gear bail and universal joint angles. The $\{IGD\}$ frame is attached to the inner gear bail, and $\{OGD\}$ is attached to the outer gear bail, as shown in Fig. 4. Both origins are colocated with the origin of $\{3\}$. In the initial position, $\{3\}$, $\{IGD\}$, and $\{OGD\}$ are coincident. The inner gear bail rotates by angle θ_{6G} about the fixed axis $\hat{Y}_{IGD}$; the outer gear bail rotates about the fixed $\hat{X}_{OGD}$ by θ_{5G}.

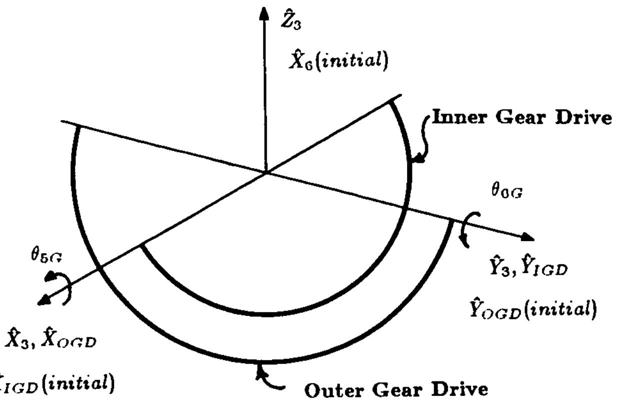


Figure 4
Definition of $\{IGD\}$ and $\{OGD\}$

From Fig. 1, the offset vector from the first to the second universal joint is a length L along $\hat{X}_6$ (denoted $\{{}^3P_8\}$). The moving axes $\hat{X}_{IGD}$ and $\hat{Y}_{OGD}$ are perpendicular to $\{{}^3P_8\}$ for any wrist motion.

$$\hat{X}_6 = \frac{\hat{X}_{IGD} \times \hat{Y}_{OGD}}{\left|\hat{X}_{IGD} \times \hat{Y}_{OGD}\right|} \tag{27}$$

The terms for Eq. 27 follow, expressed in $\{3\}$.

$${}^3\hat{X}_{IGD} = [{}^3_{IGD}R]\ {}^{IGD}\hat{X}_{IGD} = \begin{Bmatrix} c_{6G} \\ 0 \\ -s_{6G} \end{Bmatrix} \tag{27a}$$

$${}^3\hat{Y}_{OGD} = [{}^3_{OGD}R]\ {}^{OGD}\hat{Y}_{OGD} = \begin{Bmatrix} 0 \\ c_{5G} \\ s_{5G} \end{Bmatrix} \tag{27b}$$

$${}^3\hat{X}_6 = [{}^3_6R]\ {}^6\hat{X}_6 = \begin{Bmatrix} K1 \\ K2 \\ c_5c_6 \end{Bmatrix} \tag{27c}$$

Substituting these terms yields three scalar equations relating the gear bails and universal joint angles.

$$\frac{c_{5G}s_{6G}}{M} = c_4s_6 + s_4s_5c_6 = K1 \tag{28a}$$

$$\frac{-s_{5G}c_{6G}}{M} = s_4s_6 - c_4s_5c_6 = K2 \tag{28b}$$

$$\frac{c_{5G}c_{6G}}{M} = c_5c_6 \tag{28c}$$

$$M = \sqrt{cos^2\theta_{5G} + sin^2\theta_{5G}cos^2\theta_{6G}} \tag{28d}$$

Equations 28a - 28c are used to find the mappings $F2$ and $I2$.

The angle θ_4 does not have a gear bail; the $F2$ mapping is identity.

$$\theta_4 = \theta_{4G} \tag{29}$$

Using Eq. 29 in Eqs. 28a, b, and c, the following equations result.

$$A = \frac{c_{5G}s_{6G}}{Mc_{4G}} = s_6 + t_{4G}s_5c_6 \tag{30a}$$

$$B = \frac{-s_{5G}c_{6G}}{Mc_{4G}} = t_{4G}s_6 - s_5c_6 \tag{30b}$$

$$C = \frac{c_{5G}c_{6G}}{M} = c_5c_6 \tag{30c}$$

The $sin\theta_5$ term is eliminated from Eqs. 30a and b to solve for θ_6.

$$\theta_6 = sin^{-1}(u) \qquad u = \left[\frac{A + Bt_{4G}}{1 + t_{4G}^2}\right] \tag{31}$$

The inverse sine function yields θ_6 and $\pi - \theta_6$; the latter is out of the motion range of the Omni-Wrist. With θ_6 known, the solution for θ_5 comes from Eq. 30c.

$$\theta_5 = cos^{-1}(v) \qquad v = \left[\frac{C}{c_6}\right] \tag{32}$$

The inverse cosine function solution is $\pm\theta_5$. Since both results are potentially in the motion range of the Omni-Wrist, this ambiguity must be resolved by choosing the θ_5 sign which satisfies Eq. 30b. Map $F2$ is unique.

The mapping $I2$ solves for the gear bail angles given the universal joint angles. The θ_{4G} mapping is Eq. 29. The remaining gear bail angles are found by dividing Eqs. 28b and 28a by Eq. 28c.

$$\theta_{5G} = tan^{-1}(w) \qquad w = \left[\frac{-s_4s_6 + c_4s_5c_6}{c_5c_6}\right] \tag{33a}$$

$$\theta_{6G} = tan^{-1}(q) \qquad q = \left[\frac{c_4s_6 + s_4s_5c_6}{c_5c_6}\right] \tag{33b}$$

Both results from the inverse tangent function are mathematically correct, due to symmetry. However, considering angular limits of the Omni-Wrist, only quadrant I or IV results are admissible. Therefore, the inverse map $I2$ is unique.

Omni-Wrist Velocity Kinematics

Figure 5 shows the three forward and inverse maps relating the four levels of Omni-Wrist velocity parameters. The forward velocity problem finds the cartesian rates given the actuator joint rates, using maps $FV1, FV2$, and $FV3$. The inverse velocity problem accepts $\{{}^8\omega_8\}$ and calculates the actuator joint rates via maps $IV1, IV2$, and $IV3$.

The velocity maps $FV3$ and $IV1$ are Eqs. 21 and 22, and Eq. 24, respectively. The remaining Omni-Wrist velocity solutions are presented below.

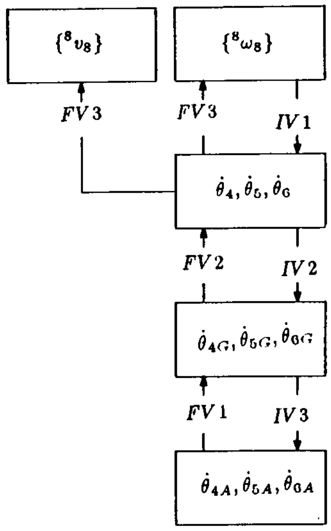
Velocity Maps $FV1$ and $IV3$

The map $FV1$ is a time derivative of Eqs. 26; $IV3$ is the inverse of Eqs. 34.

$$\dot{\theta}_{4G} = N_1\dot{\theta}_{4A} \tag{34a}$$

$$\dot{\theta}_{5G} = N_2\dot{\theta}_{5A} \tag{34b}$$

$$\dot{\theta}_{6G} = N_3\dot{\theta}_{6A} \tag{34c}$$



$\{^8v_8\}$
$\{^8\omega_8\}$
$FV3$
$FV3$
$IV1$
$\dot\theta_4, \dot\theta_5, \dot\theta_6$
$FV2$
$IV2$
$\dot\theta_{4G}, \dot\theta_{5G}, \dot\theta_{6G}$
$FV1$
$IV3$
$\dot\theta_{4A}, \dot\theta_{5A}, \dot\theta_{6A}$

Figure 5
Velocity Mappings

Velocity Maps $FV2$ and $IV2$

The map $FV2$ is a time derivative of $F2$, Eqs. 29, 31, and 32. The angular rate $\dot\theta_6$ is required for the $\dot\theta_5$ calculation.

$$\dot\theta_4 = \dot\theta_{4G} \tag{35}$$

$$\begin{aligned}
\dot\theta_6 &= \frac{1}{\sqrt{1-u^2}}\frac{du}{dt} \\
\frac{du}{dt} &= (Bc2\theta_{4G} - As2\theta_{4G})\dot\theta_{4G} + c_{4G}^2\dot A + c_{4G}s_{4G}\dot B \\
\dot A &= Q\left[t_{4G}c_{5G}s_{6G}\dot\theta_{4G} - s_{5G}s_{6G}\dot\theta_{5G} + c_{5G}c_{6G}\dot\theta_{6G} - \frac{c_{5G}s_{6G}}{M}\dot M\right] \\
\dot B &= Q\left[-t_{4G}s_{5G}c_{6G}\dot\theta_{4G} - c_{5G}c_{6G}\dot\theta_{5G} + s_{5G}s_{6G}\dot\theta_{6G} + \frac{s_{5G}c_{6G}}{M}\dot M\right] \\
Q &= \frac{1}{Mc_{4G}} \\
\dot M &= \frac{-s_{5G}s_{6G}}{\sqrt{1-s_{5G}^2s_{6G}^2}}\left[c_{5G}s_{6G}\dot\theta_{5G} + s_{5G}c_{6G}\dot\theta_{6G}\right]
\end{aligned} \tag{36}$$

$$\begin{aligned}
\dot\theta_5 &= \frac{-1}{\sqrt{1-v^2}}\frac{dv}{dt} \\
\frac{dv}{dt} &= \frac{1}{c_6}\left[Ct_6\dot\theta_6 + \dot C\right] \\
\dot C &= \frac{-1}{M}\left[s_{5G}c_{6G}\dot\theta_{5G} + c_{5G}s_{6G}\dot\theta_{6G} + \frac{c_{5G}c_{6G}}{M}\dot M\right]
\end{aligned} \tag{37}$$

The inverse map $IV2$ is a time derivative of $I2$, Eqs. 33*a* and 33*b*. The mapping for $\dot\theta_{4G}$ is Eq. 35.

$$\begin{aligned}
\dot\theta_{5G} &= \frac{1}{1+w^2}\frac{dw}{dt} \\
\frac{dw}{dt} &= \frac{-1}{c_5}(c_4t_6 + s_4s_5)\dot\theta_4 + \frac{1}{c_5^2}(c_4 + s_4s_5t_6)\dot\theta_5 - \frac{1}{c_5c_6^2}(s_4)\dot\theta_6
\end{aligned} \tag{38}$$

$$\begin{aligned}
\dot\theta_{6G} &= \frac{1}{1+q^2}\frac{dq}{dt} \\
\frac{dq}{dt} &= \frac{1}{c_5}(c_4s_5 - s_4t_6)\dot\theta_4 + \frac{1}{c_5^2}(s_4 + c_4s_5t_6)\dot\theta_5 + \frac{1}{c_5c_6^2}(c_4)\dot\theta_6
\end{aligned} \tag{39}$$

The derivatives Eqs. 36, 38, and 39 hold for the angle range $\frac{-\pi}{2}$ to $\frac{\pi}{2}$. The sign of $\dot\theta_5$ in Eq. 37 is positive when $-\pi < \theta_5 < 0$.

EXAMPLES

This section presents two examples to demonstrate the equations derived in this paper. The first example deals with the forward and inverse position and velocity problems for the general double universal joint robot wrist mechanism. The second presents forward and inverse position and velocity results for the Omni-Wrist. The dimensions used in this section are *mm, degrees*, $\frac{mm}{s}$, and $\frac{rad}{s}$.

Example 1
Forward Position

Given $\theta_4 = 120.0°, \theta_5 = -25.0°, \theta_6 = 10.0°$, and $L = 41$, $[^3_8T]$ is calculated using Eq. 3.

$$[^3_8T] = \begin{bmatrix} -0.494 & -0.798 & -0.345 & -18.3 \\ -0.452 & -0.103 & 0.886 & -2.4 \\ -0.743 & 0.593 & -0.310 & 36.6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

Inverse Position

Given $[^3_8R]$ from Eq. 40 three universal joint angles are calculated with Eqs. 14, 16, and 17; the four solutions are formed from Table II.

Table III Inverse Position Solutions

Solution	θ_4	θ_5	θ_6
1	120.0	-25.0	10.0
2	120.0	-25.0	190.0
3	120.0	155.0	-10.0
4	120.0	155.0	170.0

Forward Velocity

Given $\dot\theta_4 = 1.0, \dot\theta_5 = 2.0, \dot\theta_6 = 3.0$, and $L = 41$, $\{^8\omega_8\}$ and $\{^8v_8\}$ are calculated using Eqs. 21 and 22.

$$\{^8\omega_8\} = \begin{Bmatrix} 4.4 \\ 3.7 \\ 3.6 \end{Bmatrix} \qquad \{^8v_8\} = \begin{Bmatrix} -80.0 \\ -10.0 \\ 100.0 \end{Bmatrix} \tag{41}$$

Inverse Velocity (Resolved Rate)

Given $\{^8\omega_8\}$ from Eq. 41, $\dot\theta_4 = 1.0, \dot\theta_5 = 2.0, \dot\theta_6 = 3.0$, are calculated using Eq. 24.

Example 2
Forward Position

Given the actuator angles, the gear bail angles, universal joint angles, and $[^3_8T]$ are calculated successively, using maps $F1, F2$, and $F3$. Example 1 presents the $F3$ result.

$$\begin{array}{ccccc} \{\theta_A\} & \overset{F1}{\Longrightarrow} & \{\theta_G\} & \overset{F2}{\Longrightarrow} & \{\theta\} \\ \begin{Bmatrix} -21060.0 \\ 959.0 \\ 5852.0 \end{Bmatrix} & & \begin{Bmatrix} 120.0 \\ 3.7 \\ -26.6 \end{Bmatrix} & & \begin{Bmatrix} 120.0 \\ -25.0 \\ 10.0 \end{Bmatrix} \end{array} \qquad (42)$$

Inverse Position

Given $[^3_8R]$ from Eq. 40, the universal joint, gear bail, and actuator angles are calculated using $I1, I2$, and $I3$. Example 1 presents $I1$. Considering angular limits, only the first solution in Table III is reachable. The inverse maps $I2$ and $I3$ are the reverse of maps $F2$ and $F1$ in Eq. 42, respectively.

Forward Velocity

Given the actuator rates, the gear bail, universal joint, and cartesian rates are calculated with the mappings $FV1, FV2$, and $FV3$. Example 1 gives $FV3$.

$$\begin{array}{ccccc} \{\dot{\theta}_A\} & \overset{FV1}{\Longrightarrow} & \{\dot{\theta}_G\} & \overset{FV2}{\Longrightarrow} & \{\dot{\theta}\} \\ \begin{Bmatrix} -175.5 \\ -907.2 \\ -88.0 \end{Bmatrix} & & \begin{Bmatrix} 1.0 \\ -3.5 \\ 0.4 \end{Bmatrix} & & \begin{Bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{Bmatrix} \end{array} \qquad (43)$$

Inverse Velocity (Resolved Rate)

Given $\{^8\omega_8\}$ from Eq. 41, the universal joint, gear bail, and actuator rates are found, using $IV1, IV2$, and $IV3$. Example 1 presents $IV1$. The mappings $IV2$ and $IV3$ are the reverse of $FV2$ and $FV1$ in Eq. 43, respectively.

CONCLUSION

This paper presents kinematic equations for control of a double universal joint robot wrist. The forward and inverse position and velocity problems were solved. The Omni-Wrist equations were developed in detail. This wrist has four levels of kinematic parameters. Three forward and inverse position and velocity maps relating these parameters were presented. These equations relate the hand coordinate frame to the wrist base coordinate frame, and are sufficient for controlling the wrist standing alone. All pertinent kinematic equations were derived; any specific control algorithm will not require all of the equations. All Omni-Wrist solutions are unique. The Omni-Wrist is completely singularity-free throughout its range of motion.

The equations of this paper have been verified by computer simulation. As demonstrated by the examples, the inverse solutions validate the forward solutions. Experimental work using the Omni-Wrist is planned to further validate the equations.

The offset, L, between the two universal joints complicates the inverse kinematics problems when the double universal joint robot wrist is attached to a manipulator arm. The wrist coordinate frames are not all colocated, which prevents decoupling of the hand coordinate frame position and orientation. For a three degree of freedom manipulator arm carrying the double universal robot wrist, the inverse position problem involves six transcendental equations, coupled in the six unknowns. The associated Jacobian matrix is fully populated, which means the hand linear velocity depends on the wrist rates in addition to the first three joint rates. The kinematics of a manipulator using the double universal joint robot wrist is a subject for future research.

REFERENCES

Barker, L.K., "Theoretical Three- and Four-Axis Gimbal Robot Wrists", NASA Technical Paper 2564, May 1986.

Craig, J.J., *Introduction to Robotics: Mechanics and Control,* Addison Wesley Publishing Co., Reading, MA, 1988.

Mabie, H.H., and Reinholtz, C.F., *Mechanisms and Dynamics of Machinery,* John Wiley & Sons, New York, 1987.

McKinney, W.S.,"Kinematics of Hooke Universal Joint Robot Wrists", NASA Technical Memorandum 100567, June 1988.

Milenkovic, Veljko, "New Nonsingular Robot Wrist Design", *Robots 11 Conference Proceedings RI/SME,* Chicago, IL, April 1987, pp. 13.29-13.42.

Rosheim, M.E., *Robot Wrist Actuators,* John Wiley & Sons, New York, 1989.

Rosheim, M.E., "Singularity-Free Hollow Spray Painting Wrists", *Robots 11 Conference Proceedings RI/SME,* Chicago, IL, April 1987, pp. 13.7-13.28.

Rosheim, M.E., "Four New Robot Wrist Actuators", *Robots 10 Conference Proceedings RI/SME,* Chicago, IL, April 1986, pp. 8.1-8.45.

Trevelyan, J.P., et. al., "ET: A Wrist Mechanism Without Singular Positions", *The International Journal of Robotics Research,* Winter, 1986, pp. 71-85.

N91-20646

PERFORMANCE CAPABILITIES OF A JPL DUAL-ARM ADVANCED TELEOPERATION SYSTEM

Z.F. Szakaly and A.K. Bejczy
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

The system comprises a) two PUMA 560 robot arms, each equipped with the latest JPL-developed smart hands which contain 3D force/moment and grasp force sensors, b) two general-purpose force-reflecting hand controllers, c) a NS32016 microprocessors based distributed computing system together with JPL-developed universal motor controllers, d) graphics display of sensor information, e) capabilities for time-delay experiments, and f) automatic data recording capabilities. Several different types of control modes are implemented on this system using different feedback control techniques. This paper describes some of the control modes and the related feedback control techniques, and reports on the achievable control performance for tracking position and force trajectories. The interaction between position and force trajectory tracking is illustrated. The best performance is obtained by using a novel, task-space error feedback technique.

INTRODUCTION

The JPL dual-arm advanced teleoperation hardware system is shown in Figure 1. It employees a novel generalized bilateral force-reflecting control method for manual control of remote manipulators. The novel features of this control method are the following: (1) The master controller is a general purpose Force-Reflecting Hand Controller (FRHC), not a replica of any slave arm. It can be used to control different robot arms through the appropriate kinematic transformations. (2) Force reflection to the operator's hand is referenced to a three-d.o.f. force-torque sensor mounted to the base of the robot hand. (3) The control system is based on distributed computing; it uses two computing nodes for control and information display: one at the control station (FRHC) site and one at the remote robot site.

The system permits a spectrum of operations between full manual, "shared" manual and automatic, and full automatic (called "traded") control, and can be operated with variable active compliance referenced to force-torque sensor. Shared control is implemented by freezing the data output of the master controller (FRHC) in some task space coordinates which are selectable by the operator from a menu. Motion in the frozen task space coordinates can then be controlled by a computer algorithm which can be referenced to force-torque or to some other (e.g., proximity) sensor information.

The overall hardware system, electronic architecture, software system including control modes, control algorithms and the software development system, the real-time graphics (preview and predictive displays) including force-torque sensor data displays, and time-delay simulation capabilities are described in previous publications [1 and 2] which contain further references on other hardware and software details. The "smart hands" attached to the robot arms also represent special features of the JPL dual-arm advanced teleoperation system. The Model B and Model C "smart hands" (shown in Fig. 1) mechanical and electronic details are described in [3].

The purpose of this paper is to describe in detail the currently available control modes and the related feedback control techniques implemented on the JPL dual-arm advanced teleoperation system, and to report on the achievable control performance for tracking position and force trajectories. In the description of performance results, emphasis is given to comparing position and force tracking performance with and without Cartesian servo.

Cartesian (or task-space) servo is a novel feedback technique to correct in the time continuum for position errors. In this technique, task space errors are computed from actual joint space values and actual task space commands. (Eventually, task space errors can be measured directly when such measurement system becomes available.) This novel Cartesian error feedback technique can be applied either to FRHC manual trajectory commands or to trajectory commands generated by an algorithm. Here we formulated and used a novel trajectory generator algorithm. This novel trajectory generator algorithm directly acts on task space position commands without a time-based polynomial decomposition of position commands into joint space or task space trajectories. The velocity (when it is not a constant) and, implicitly, the change of velocity in this novel trajectory generator algorithm follows the profile of harmonic functions. Hence the name: Harmonic Motion Generator (HMG).

First we describe the control modes followed by a discussion of performance data.

CONTROL MODES

The overall data flow diagram of the JPL advanced teleoperation system (for a single arm, for the sake of simplicity) is shown in Fig. 2. It is noted that the computing architecture of this system is a fully synchronized pipeline, where the local servo loops at both the control station and the remote manipulator nodes operate at 1000 Hz rate. The end-to-end bilateral (i.e., force-reflecting) control loop operates at a 200 Hz rate as indicated in the computation system timing diagram, Fig. 3. More on the computational system critical path functions and performance can be found in [4].

The actual data flow depends on the control mode chosen. The different selectable control modes are the following:

- Freeze mode
- Neutral mode
- Current mode
- Joint mode
- Task mode

In Freeze mode the brakes of joints 1,2,3 are locked, the motors are turned off. Joints 4,5,6 are servoed to maintain their last positions. This mode is primarily used when the robot is not needed for a short period of time but turning it off is not desired.

In Neutral mode all position gains are set to 0, gravity compensation is active to prevent the robot from falling down. In this mode the user can manually move the robot to any position and it will stay there.

In Current mode the six motor currents are directly commanded by the data coming in from the fiber optic link. This mode exists for debugging only.

In Joint mode the hand controller axes control individual motors of the robot. The correspondence is set up such that in the most common lower elbow/ inverted wrist configuration the joint mode controls the robot in the naturally expected directions i.e., similar to task mode.

In Task mode the inverse kinematic transformation is performed on the incoming data, the hand controller controls the end effector tip along the three Cartesian and pitch, yaw and roll axes. This mode is the most frequently used for task execution or experiments, and this is the one shown explicitly in Fig. 2.

The format of the data packet transmitted to the robot side is the same in all modes. The header byte defines the mode the robot should be in. This is followed by the six motion command bytes, the grasping force command and a checksum. If the mode byte changes the robot waits until the new mode has been stable for 1000 servo loops or one second. After one second the new mode becomes active.

The data packet coming back from the robot is always formatted the same way independent of what mode the robot is in. The following data is transmitted to the local site:

- Six words of force sensor data
- Grasping force and finger opening
- Robot joint position
- End effector tip Cartesian positions

The control system on the remote site is designed to prevent sudden robot motions. The motion commands received by the fiber optic link are incremental, they are added to the current parameter under control. Sudden large motions are also prevented in case of mode changes. This necessitates proper initialization of the inverse kinematics software at the time of the mode transition. The current Cartesian coordinates from the forward kinematics are input into the inverse one. Besides this the configuration parameters such as upper or lower elbow, normal or inverted wrist have to be correctly initialized.

The data flow diagram shown in Fig. 2 illustrates the organization of several servo loops in the system. The innermost loop is the position control servo of the robot side. This servo uses a PD control algorithm, where the damping is purely a function of the robot joint velocities. The incoming data to this servo is the desired robot trajectory described as a sequence of points at 1 mSec intervals. This joint servo is augmented by the gravity compensation routine to prevent the weight of the robot from causing joint positioning error. Since this servo is a first order one there will be a constant position error that is proportional to the joint velocity.

In basic Cartesian control mode the data from the fiber optic link is integrated first and added to the desired Cartesian position. From this the inverse kinematics generates the desired joint positions. The joint servo moves the robot to this position. From the actual joint position the forward kinematics computes the actual Cartesian positions. The force torque sensor data and the actual positions are fed back to the hand controller side to provide force feedback.

This basic mode can be augmented by the addition of the following:

- Compliance control,
- Cartesian servo,
- Sticktion, friction compensation.

Figure 4 Specifically shows the compliance control and Cartesian servo augmentations.

There are two forms of compliance, integrating and spring type (see Fig. 5). In integrating compliance the velocity of the robot end effector is proportional to the force felt in the corresponding direction. To eliminate drift a dead-band is used. The zero velocity band does not have to be a zero force, a force offset may be used. Such a force offset is used if, for example, we want to push against the task board at a given force while moving along other axes. Any form of compliance can be selected along any axis independently.

In case of the spring type compliance the robot position is proportional to the sensed force. This is similar to a spring centering action. The velocity of the robot motion is limited in both the integrating and spring cases.

There is a wide discrepancy between the robot response bandwidth and the force readings. The forces are read at a 1000 Hz sampling rate although the hand is capable of delivering more than 5000 samples per second. The robot motion command has an output response at a 5 Hz bandwidth. To generate smooth compliance response, the force readings go through two subsequent filters. The first one is a simple averaging of ten force readings. This average is called 100 Hz force and is computed at a 100Hz rate. From this 100 Hz force a 5 Hz force is computed by a first order low pass filter. This 5 Hz force reading is also computed at a 100 Hz rate. The 5 Hz force is used for compliance computations. The subsequent equations define the force filters and the compliance control algorithms.

<u>Force Filter</u>:

Input F_{1000}: Force at 1 KHz

F_{100}: Force at 100 Hz computed as

$$F_{100}(t) = \frac{1}{10}[F_{1000}(t) + F_{1000}(t-1) + \ldots + F_{1000}(t-9)]$$

F_{100} is computed at 100 Hz

F_5: Force at 5 Hz computed as

$$F_5(t) = F_5(t-1) + K_F[F_{100}(t) - F_5(t-1)],$$

$$K_F = \frac{1}{20}$$

F_5 is also computed at 100 Hz

<u>Compliance Control</u>: operates by modifying Cartesian set point X_S

$$X_{S2} = \underbrace{X_{S1} + K_I(F_{5X} - S_{IX})}_{\text{Integrator}} + \underbrace{K_S\{F_{5X}(t) - S_{SX}(t) - [F_{5X}(t-1) - S_{SX}(t-1)]\}}_{\text{Spring}}$$

K_I	: integrating constant
K_S	: spring constant
X_{S1}	: X setpoint coming from hand controller
S_{IX}	: X integrating force setpoint
S_{SX}	: X spring force setpoint

It is interesting to observe the similarities and differences between averaging and a low pass filter (see also Fig. 6). In order to average them we have to store the ten previous force readings. For the low pass filter a single stored variable is adequate. The step input transfer function of the averaging filter is a linearly increasing output (or more exactly ten equal steps). The same function for the low pass filter is one that exponentially approaches the steady state output value (i.e., the steps become smaller and smaller in time). In terms of filtering, the two have similar effects on the signal, but low pass filtering requires much less memory and computations.

As shown in Fig. 4, the Cartesian servo acts on task space (X,Y,Z, pitch, yaw, roll) errors directly. These errors are the difference between desired and actual task space values. The actual task space values are computed from the forward kinematic transformation of the actual joint positions. This error is then added to the new desired task space values before the inverse kinematic transformation determines the new joint position commands from the new task space commands.

TRAJECTORY GENERATOR

A trajectory generator algorithm was formulated based on observations of profiles of task space trajectories generated by the operators manually through the FRHC. Three important features were observed in hand-generated task space trajectory profiles: (1) The operators always generated trajectories as a function of the relative distance between start point and goal point in the task space or, in general, as a function of the present position state relative to the desired position state of the end effector in the task space. In other words, the operators manually do not generate trajectories based on time (on clock signals). (2) The velocity-position phase diagrams of motion typically resembled a harmonic (sine) function. (3) Between the start and completion phases, the operator-generated trajectories typically attained a constant velocity profile.

Based on these observations, we formulated a Harmonic Motion Generator (HMG) with a sinusoidal velocity - position phase function profile as shown in Fig. 7. The motion is parameterized by the total distance traveled, the maximum velocity, and the distance used for acceleration and deceleration. Both the accelerating and decelerating segments are quarter sine waves, with a constant velocity segment connecting them. This scheme still has a problem, the velocity being 0 before the motion starts. This problem is corrected by adding a small constant to the velocity function.

It is noted that the HMG introduced in this paper is quite different from the typical trajectory generator algorithms employed in robotics which use a polynomial position-time function. Our algorithm generates the motion as a trigonometric (harmonic) velocity versus position function. The position versus time and the corresponding velocity versus time functions generated by the HMG are shown in Fig. 8.

PERFORMANCE RESULTS

Space assembly and servicing tasks are very rich in capability requirements for a dual-arm teleoperation system. For instance, if the Solar Max Repair Mission would have been performed with a dual-arm teleoperation system, the operator(s) of the dual-arm system would have faced the following subtasks: thermal blanket removal, hinge attachment for electrical panel, opening of electrical panel, removal of electrical connectors, relining of cable bundles, replacement of electrical panel, securing parts and cables, replug of electrical connectors, closing of electrical panel, and reinstating thermal blanket. In order to perform all these subtasks, the dual-arm teleoperation system should be endowed with certain generic performance features. Such generic performance features are: move along a straight line and exert a given push force in a given direction (that is, cutting a thermal blanket by knife); hold a given force in a given direction while turning/rolling operation is being performed (that is, removal or reinstatement of panel screws); follow a given path while pulling a flexible object (that is, relining of cable bundles); etc.

Several performance experiments were carried out recently in order to evaluate position and force tracking capabilities of the JPL advanced dual-arm teleoperation system using various control modes and feedback techniques implemented in the system. The subsequent 12 figures (Figs. 10 through 21) show and summarize the performance capabilities. The reference frame in which the motion/force commands are interpreted is shown in Fig. 9.

One-Dimensional Straight Lines

Figures 10 through 12 show performance results of straight one-dimensional (X,Y, or Z) trajectory following, with and without Cartesian servo. The trajectories are commanded from the FRHC at 1 KHz increments, and servoed at the same rate at the remote manipulator. The FRHC task space commands can be true one-dimensional straight lines by inhibiting the computer reading of FRHC motion in the other two orthogonal task space directions. For instance, when commanding a horizontal Y straight line motion, the X and Z directional commands are automatically kept at zero, and servoed accordingly at the remote manipulator. That is, a one-dimensional straight line command is independent of the operator's ability to move the FRHC on a straight line. This capability is automatically guaranteed by the command/control software.

It is clear from Figs. 10 to 12 that Cartesian servo gives a superior and very satisfactory trajectory following performance over the non-Cartesian (that is, pure joint servo) performance. Indeed, it compensates very well for sticktion, friction, and for some level of uncertainties in gravity loading. It is noted that the remote manipulator was operated with about 80% gravity compensation control only and without sticktion and friction compensation.

Two-Dimensional Straight Lines

Figures 13 through 15 show performance results of two-dimensional (X-Z, Z-Y, Y-X) straight line trajectory following tasks, with and without Cartesian servo. Again, the trajectories were commanded from the FRHC at 1 KHz increments, and servoed at the same rate at the remote manipulator. It is noted that the quality of a straight line trajectory in a plane depends on the operator's ability to generate a true straight line with his hand motion in that plane. It is automatically guaranteed, however, that the trajectory command will be in the selected plane by inhibiting the computer reading of any FRHC motion perpendicular to the selected plane.

Again, it is clear from Figs. 13 to 15 that Cartesian servo yields a superior and very satisfactory trajectory following performance over the non-Cartesian (pure joint servo) performance. It compensates very well for sticktion, friction and uncertainties in gravity loading.

One-Dimensional Straight Lines With Force Command

Figures 16 through 19 show performance results for tasks of one-dimensional straight line trajectory following with the added requirement of maintaining a given force in a given direction along the straight line trajectory. Force control was automatic by selecting the "integrator" component of the compliance control algorithm (see Fig. 6 and the corresponding equations in the text) along the appropriate direction and at the appropriate level.

It is clear from Figs. 16 and 17 that Cartesian position servo considerably improves trajectory position following performance along the commanded motion direction. It is not clear, however, what is the role of Cartesian position servo along the commanded force-maintaining direction referenced to force sensor data. Theoretically, the two control loops contradict each other. In the actual performance, however, the "integrator-compliance" loop seemingly overrules the Cartesian position servo loop along the compliance axis. In any case, automatic compliance control shown very satisfactory performance within the mechanical limits (backlash, hysteresis, etc.) of the PUMA 560 manipulator.

For future applications it is recommended to disable Cartesian position servo along the commanded compliance axis and keep Cartesian position servo only acting along the axes where no force compliance is required.

Figures 18 and 19 also clearly show the output profiles of the 100 Hz and 5 Hz force-torque sensor data filters described previously and applied in the compliance control algorithm. The actual mechanical response profile of the manipulator's compliant interaction with the environment is along the 5 Hz filter trajectory.

Harmonic Motion Generator (HMG) Trajectories

Two examples are quoted here. Figure 20 illustrates the same trajectory following example which was shown in Fig. 10. There, the trajectory was generated by FRHC motion. Here, it is generated by the HMG outlined previously. Again, Cartesian position servo provides a much better trajectory following performance than the pure joint servo.

Figure 21 illustrates the same trajectory as shown above in Fig. 20 as generated by the HMG algorithm, with the additional requirement of maintaining a given force level in X direction along the Y-directional trajectory. For maintaining force, the integrator part of the automatic compliance algorithm was used. Cartesian servo was disabled along the compliance axis (X) but was retained along the other two (Y and Z) orthogonal axes. To make the task more challenging, the task board along the Y direction was disoriented by about 5 degrees relative to the nominal Y direction. That is, to maintain a constant force along the X direction while moving in the Y direction required an automatic position correction in the X direction based upon force sensing. As seen in Fig. 21, the automatic control system performed excellently.

It is noted that the example shown in Fig. 21 is equivalent to cutting a 40 cm long material with a knife with 5N cutting force automatically, and such that misalignment between cutting board and knife along the cut direction is automatically corrected based on the sensing of the required cutting force.

CONCLUSIONS

The quoted examples have shown the performance utility of (a) Cartesian position servo in trajectory following tasks and (b) automatic compliance in force following/maintaining tasks. Comparing Fig. 21 to Fig. 19, one can also conclude that for certain well-defined tasks (e.g., cutting a material), an automatic HMG combined with an automatic compliance control can give smoother results than an FRHC generated trajectory combined with automatic compliance control.

Future plans include the expansion of the quoted control capabilities formalized into easy operator menus. The capabilities will then be exercised on Solar Max Repair Mission (SMRM) tasks in realistic mission simulation settings in order to demonstrate existing and missing (or, to be improved) capabilities for space applications.

REFERENCES

[1] 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 25-27, 1989, NASA Conf. Publication 3059, pp. 649-660.

[2] Szakaly, Z., Kim, W.S., Bejczy, A.K., "Force-Reflective Teleoperated System with Shared and Compliant Control Capabilities," Proc. of the NASA Conf. on Space Telerobotics, Jan. 31, 1989, JPL Publication 89-7, Vol. IV, pp. 145-155.

[3] 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.

[4] Szakaly, Z., Fleischer, G., "JPL Advanced Teleoperation Control System Critical Path Performance," JPL Memo 3470-90-332.

ACKNOWLEGEMENTS

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

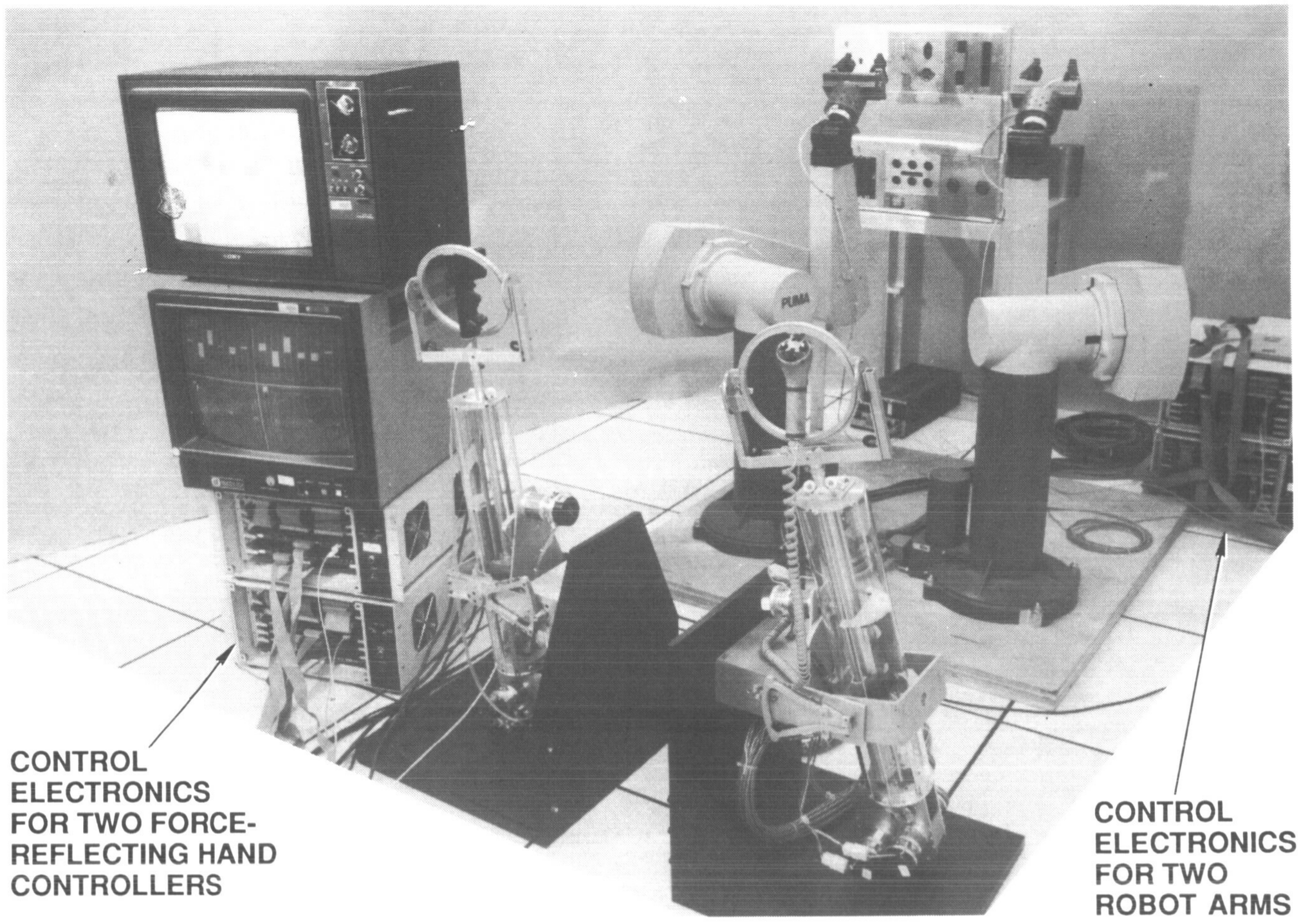

CONTROL ELECTRONICS FOR TWO FORCE-REFLECTING HAND CONTROLLERS
CONTROL ELECTRONICS FOR TWO ROBOT ARMS

Figure 1. JPL Dual-Arm Advanced Teleoperation System

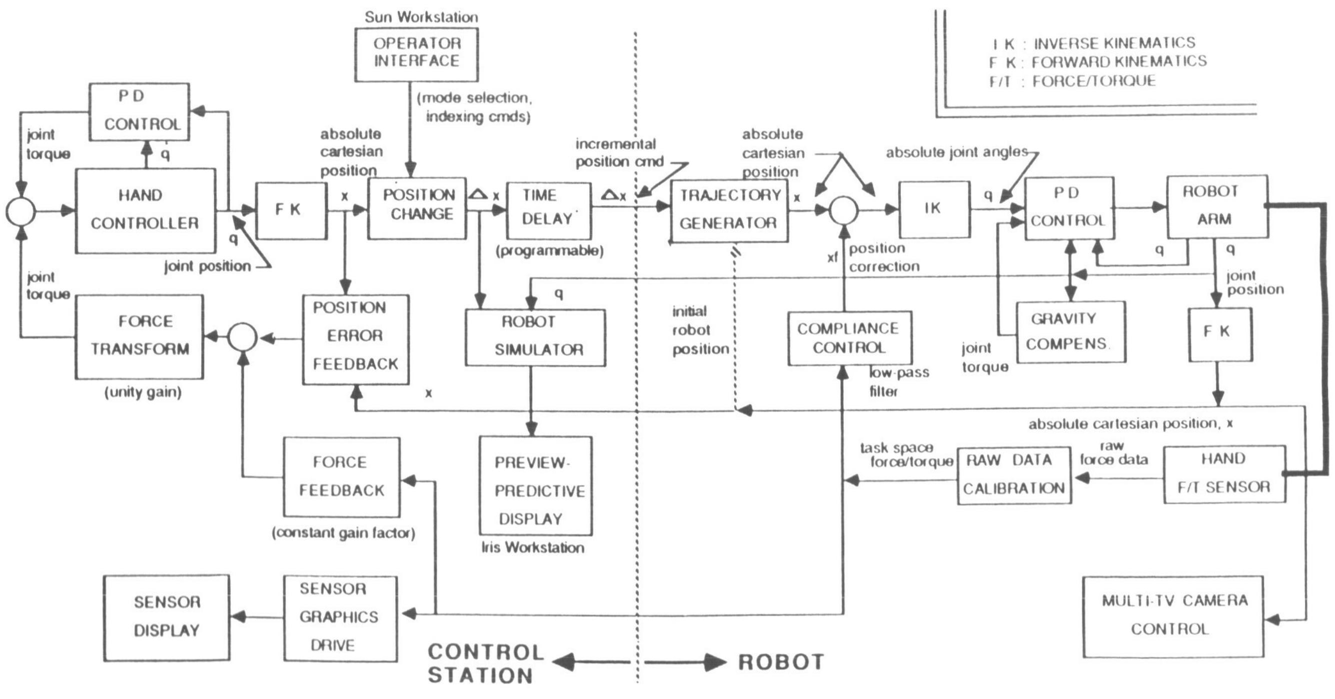


Figure 2. System Flow Diagram

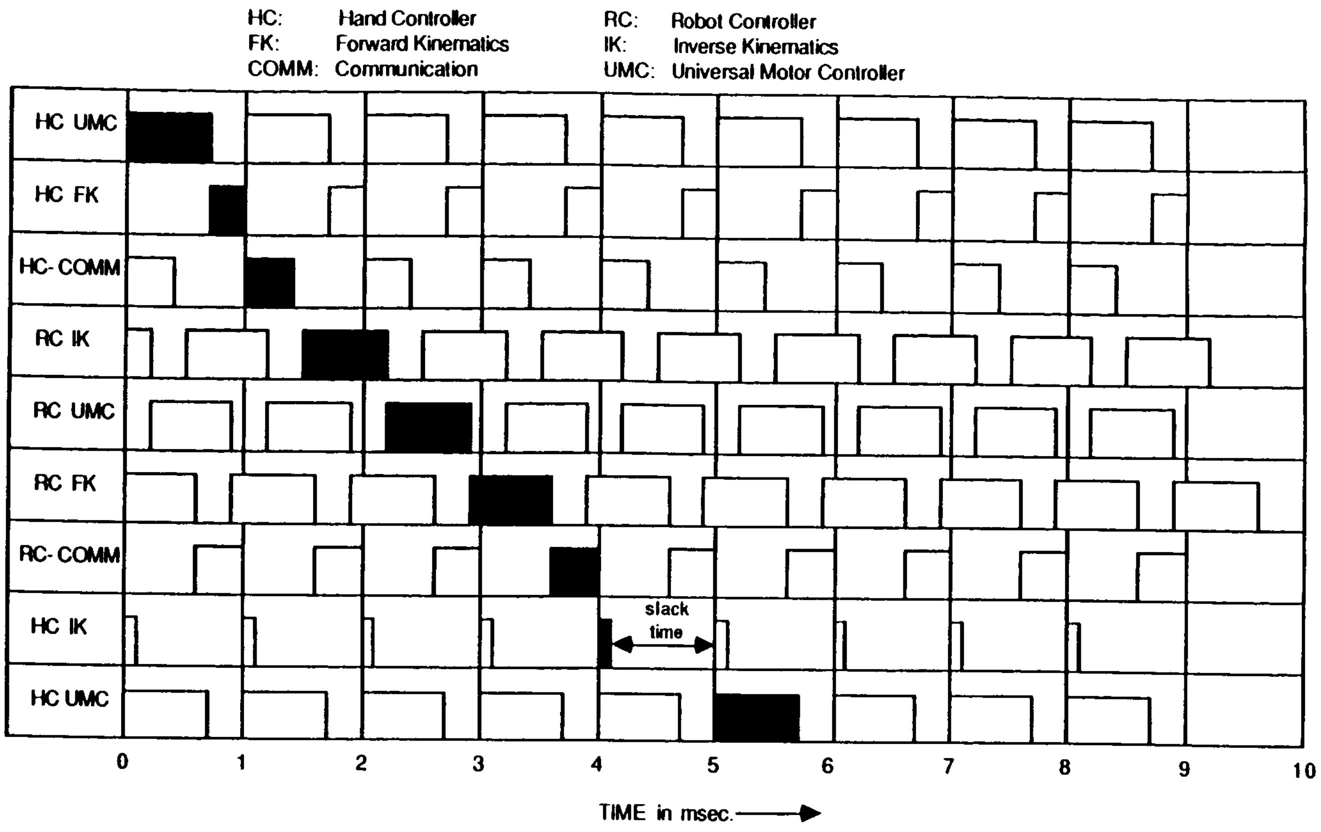


Figure 3. System Timing Diagram

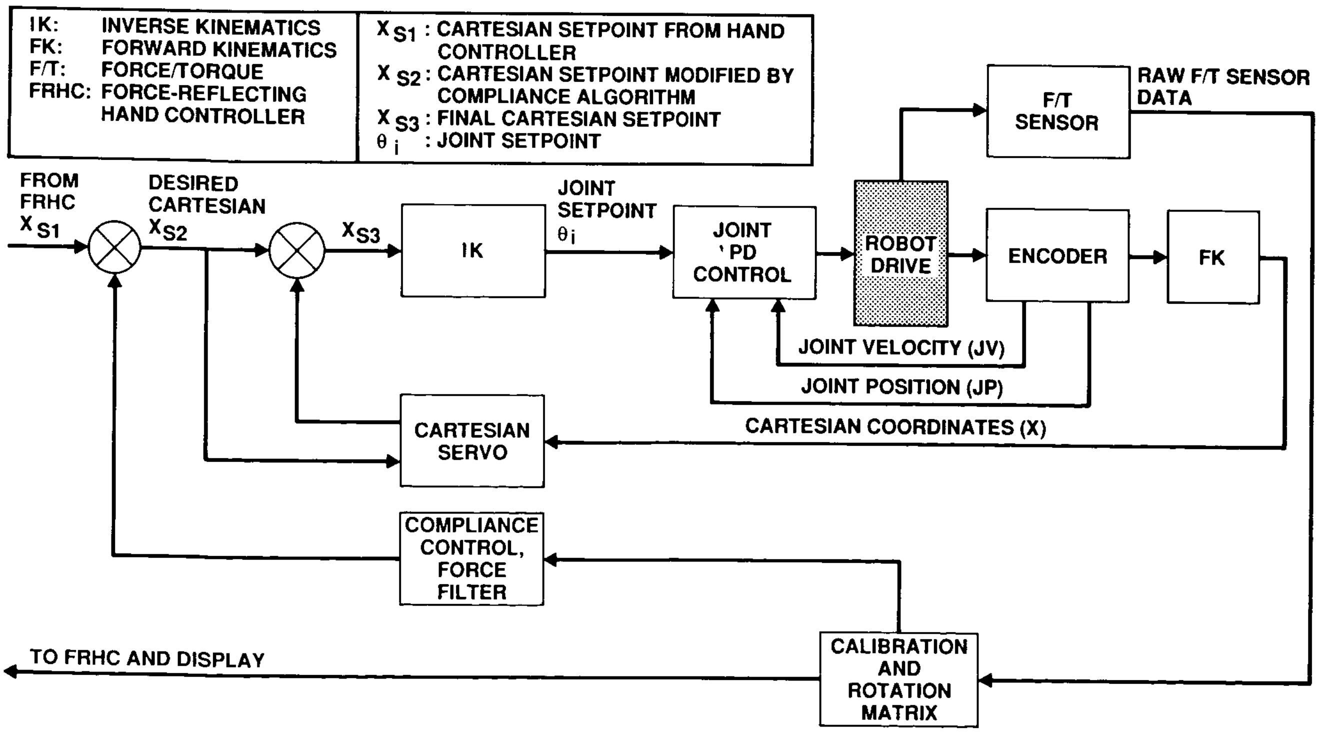

IK: INVERSE KINEMATICS
FK: FORWARD KINEMATICS
F/T: FORCE/TORQUE
FRHC: FORCE-REFLECTING HAND CONTROLLER
X_{S1} : CARTESIAN SETPOINT FROM HAND CONTROLLER
X_{S2} : CARTESIAN SETPOINT MODIFIED BY COMPLIANCE ALGORITHM
X_{S3} : FINAL CARTESIAN SETPOINT
θ_i : JOINT SETPOINT
FROM FRHC X_{S1}
DESIRED CARTESIAN X_{S2}
X_{S3}
IK
JOINT SETPOINT θ_i
JOINT PD CONTROL
ROBOT DRIVE
F/T SENSOR
RAW F/T SENSOR DATA
ENCODER
FK
JOINT VELOCITY (JV)
JOINT POSITION (JP)
CARTESIAN COORDINATES (X)
CARTESIAN SERVO
COMPLIANCE CONTROL, FORCE FILTER
CALIBRATION AND ROTATION MATRIX
TO FRHC AND DISPLAY

Figure 4. Control Schemes: Joint Servo, Cartesian Servo, Compliance Control

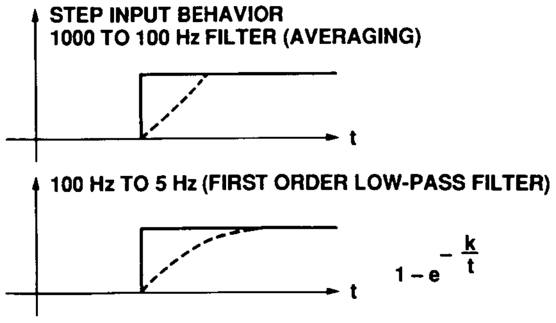


Figure 5. Force-Torque Sensor Data Filters

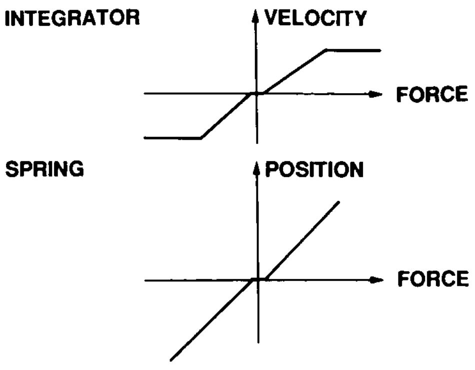


Figure 6. Compliance Components and Interpretations

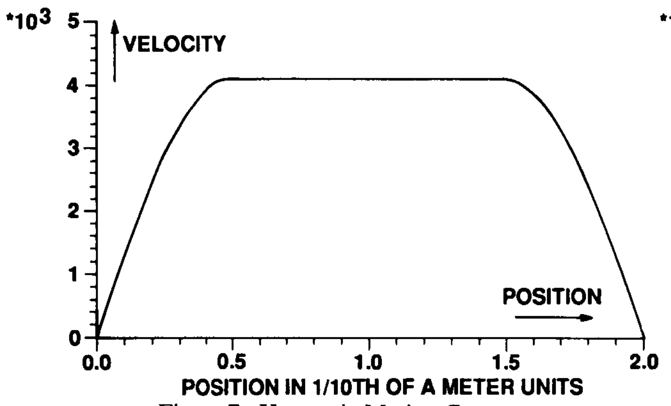


Figure 7. Harmonic Motion Generator Velocity-Position Function

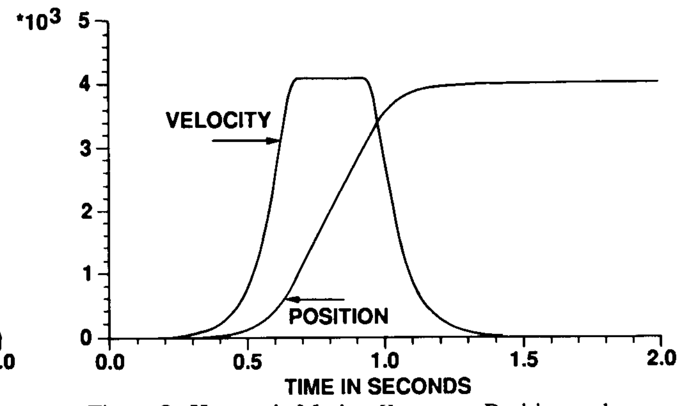


Figure 8. Harmonic Motion Generator Position and Velocity Time Functions

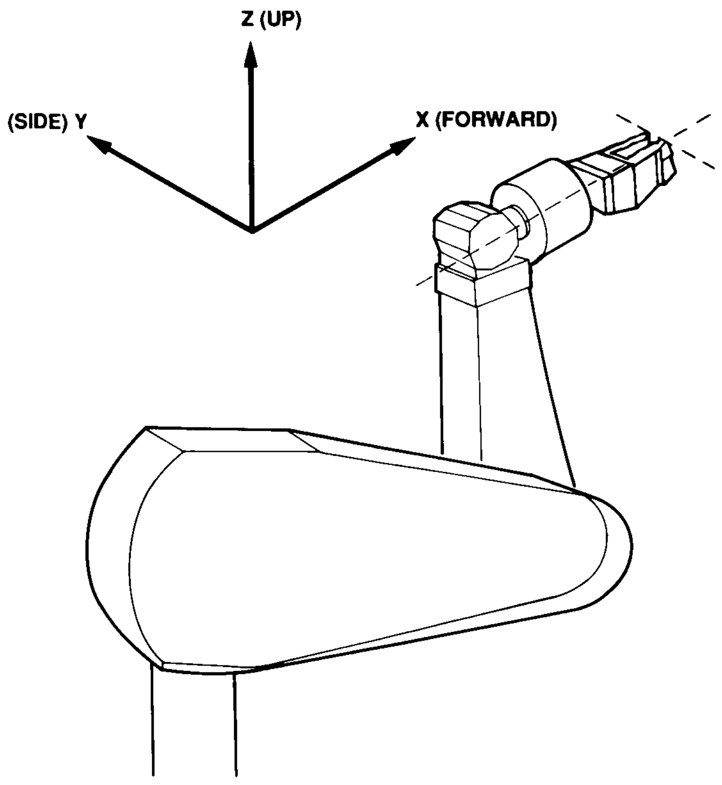


Figure 9. Reference Frame

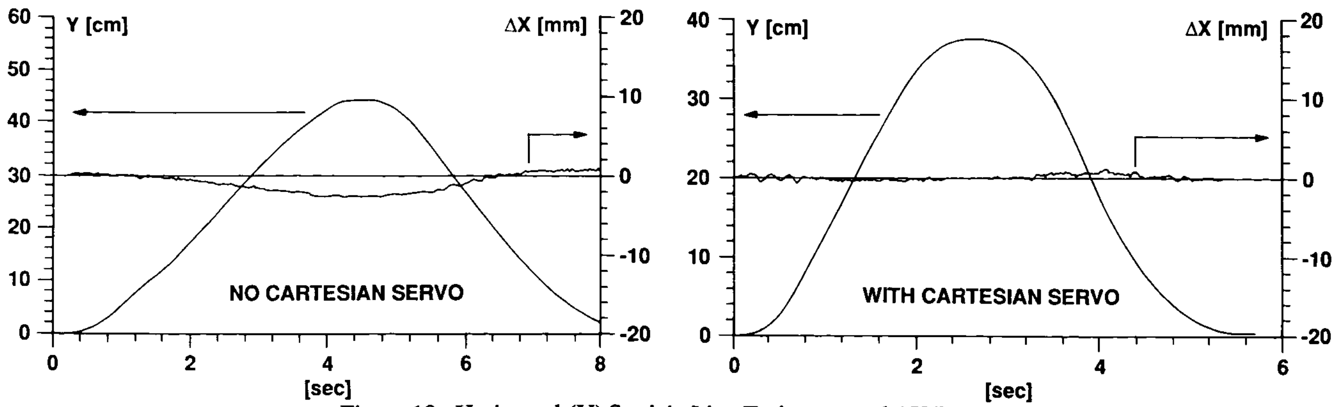


Figure 10. Horizontal (Y) Straight Line Trajectory and ΔX Error.

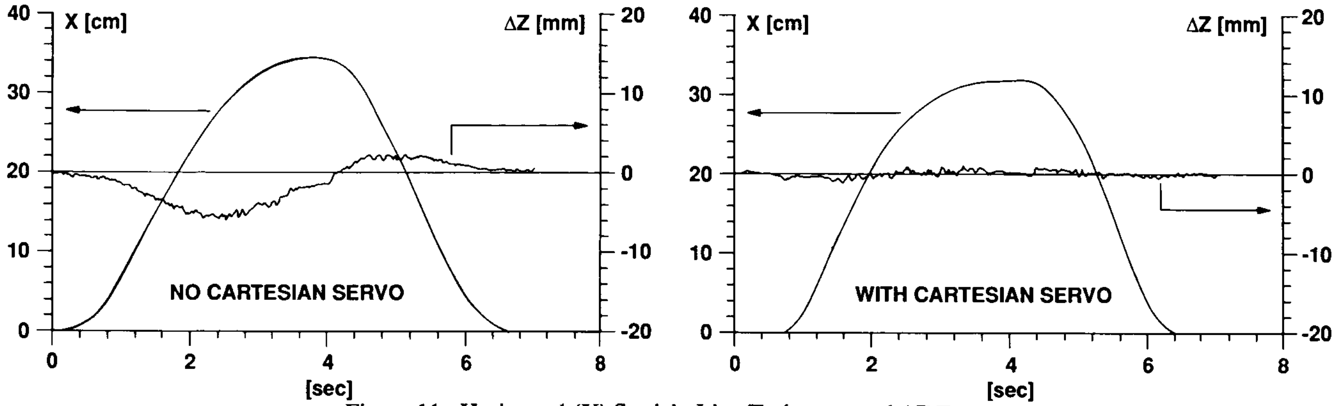


Figure 11. Horizontal (X) Straight Line Trajectory and ΔZ Error.

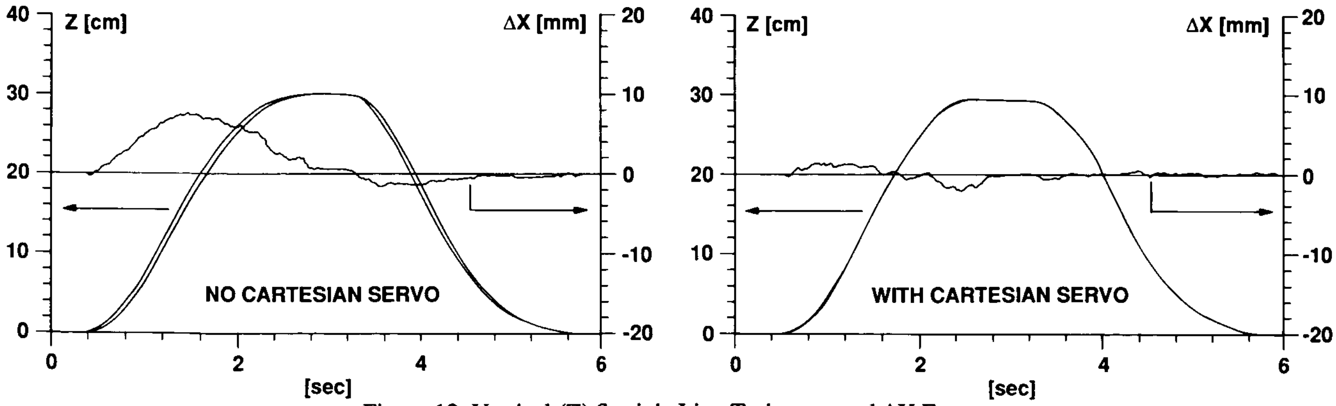


Figure 12. Vertical (Z) Straight Line Trajectory and ΔX Error.

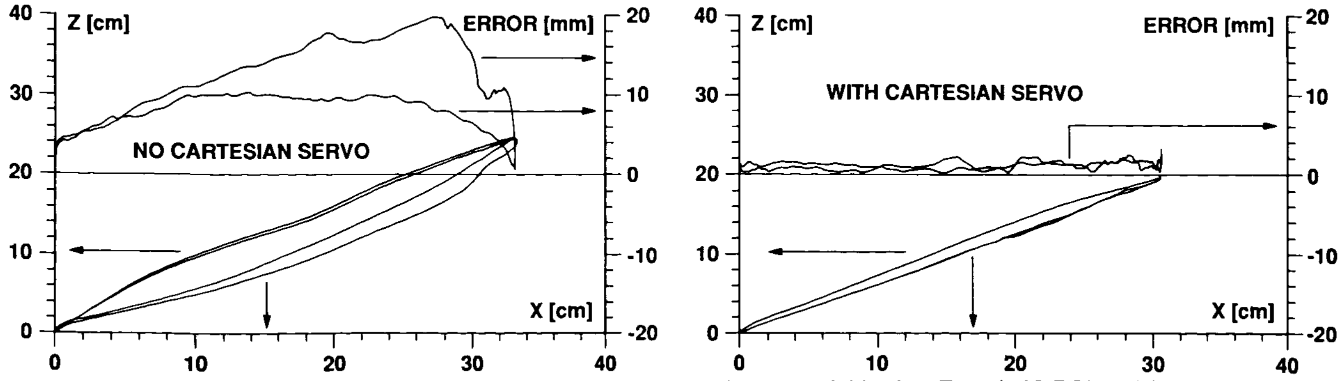


Figure 13. X-Z Plane Forward-Up/Backward-Down Trajectory and Absolute Error in X-Z Phase Plane.

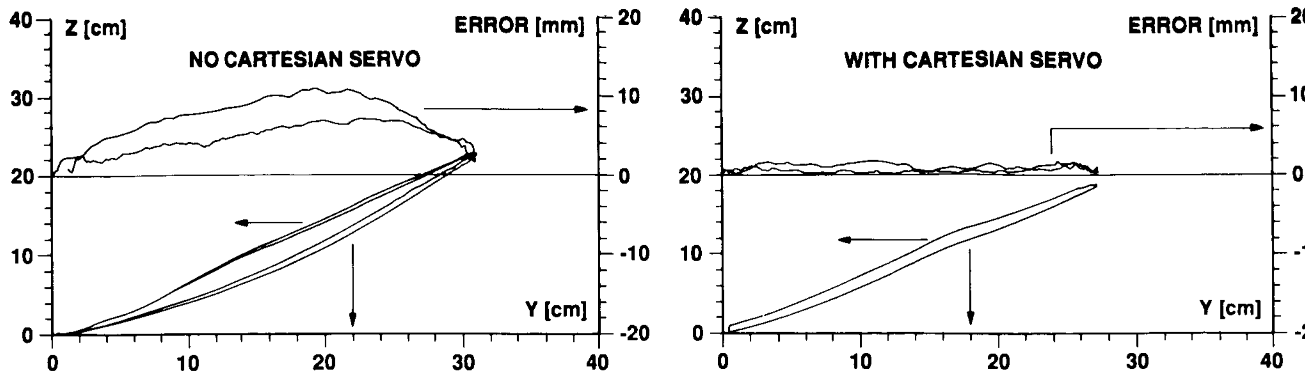


Figure 14. Y-Z Plane Right-Up/Left-Down Trajectory and Absolute Error in Y-Z Plane.

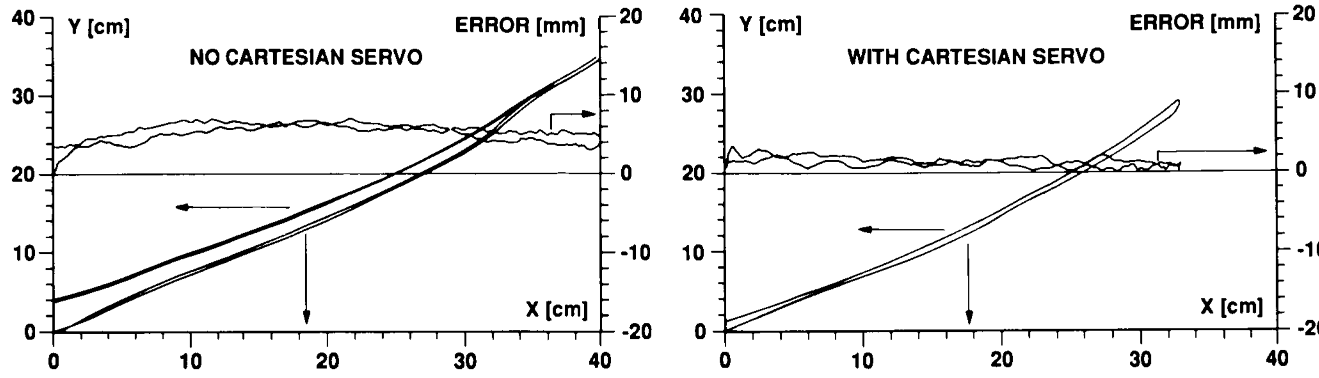


Figure 15. X-Y Plane Forward-Right/Backward-Left Trajectory and Absolute Error in X-Y Phase Plane.

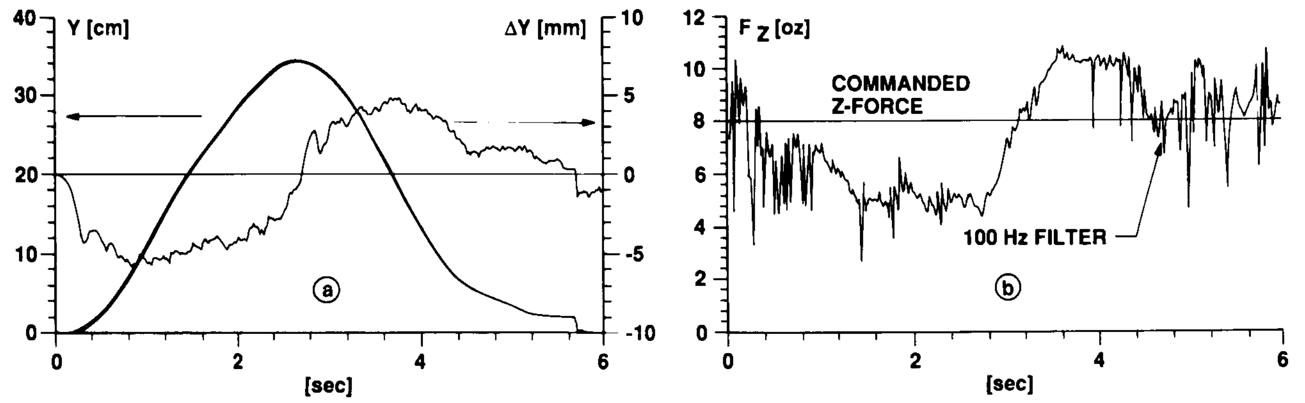


Figure 16. Horizontal (Y) Straight Line Trajectory, With Constant Z-Force Command, Without Cartesian Servo: ⓐ Position Tracking, ⓑ Force Tracking.

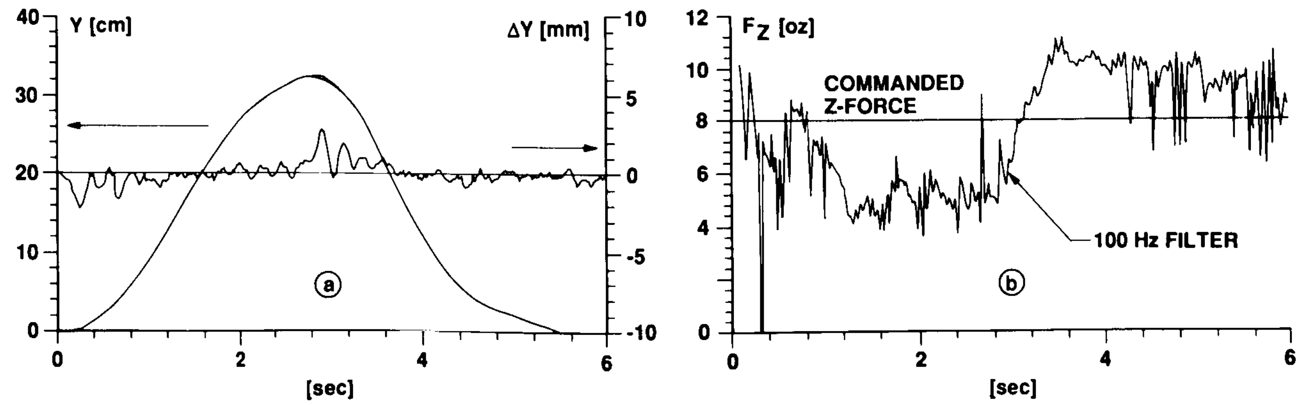


Figure 17. Content the Same as Fig.16, But With Cartesian Servo.

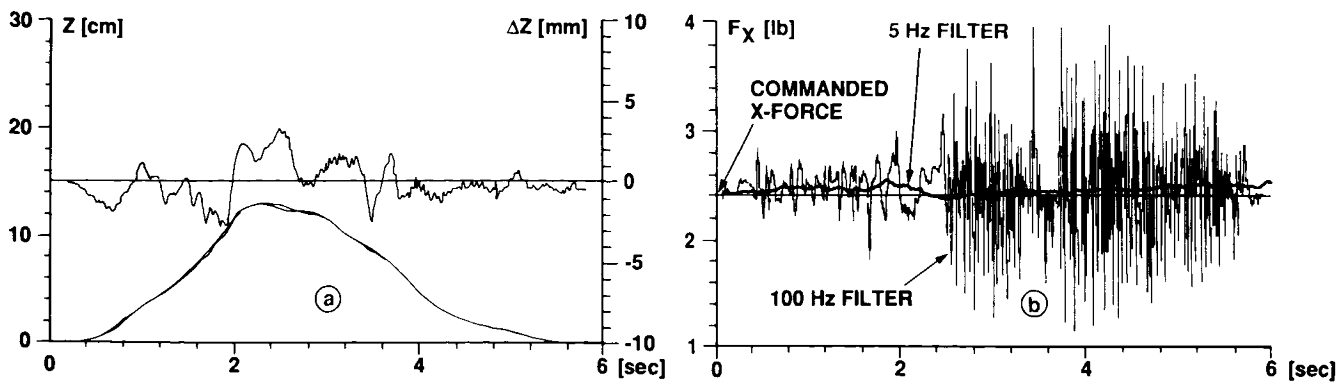


Figure 18. Vertical (Z) Straight Line Trajectory, With Constant X-Force Command, With Cartesian Servo: ⓐ Position Tracking, ⓑ Force Tracking.

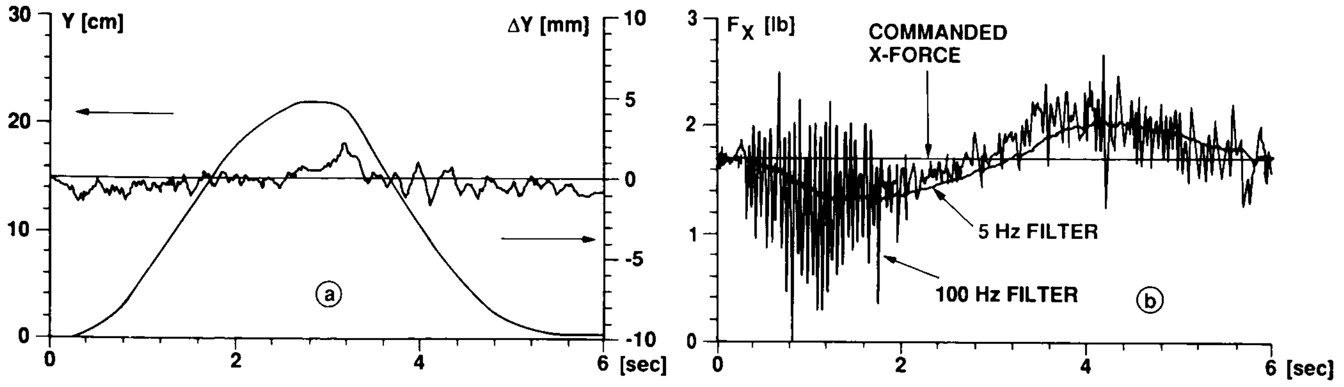


Figure 19. Horizontal (Y) Straight Line Trajecotry, With Constant X-Force Comand, With Cartesian Servo: ⓐ Position Tracking, ⓑ Force Tracking.

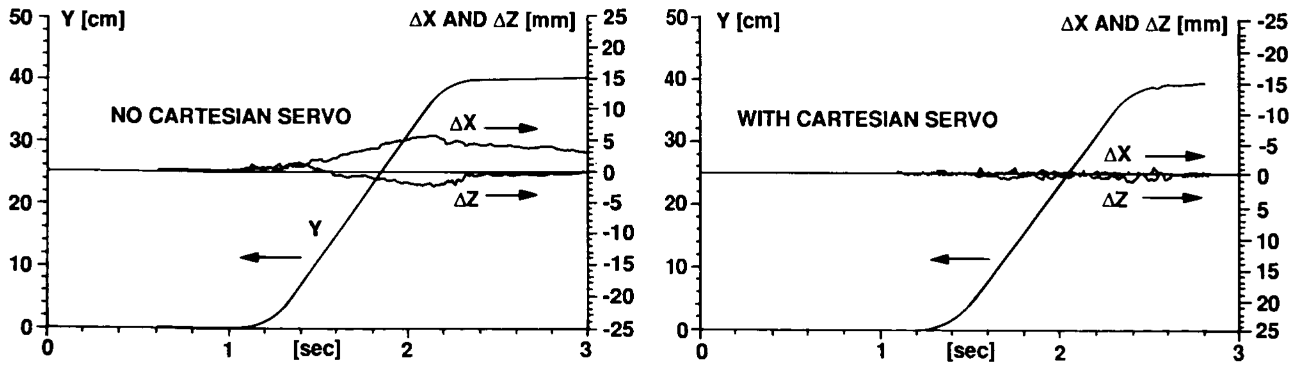


Figure 20. Horizontal (V) Straight Line Trajectory from Harmonic Motion Genrator and ΔX 2 ΔZ Errors.

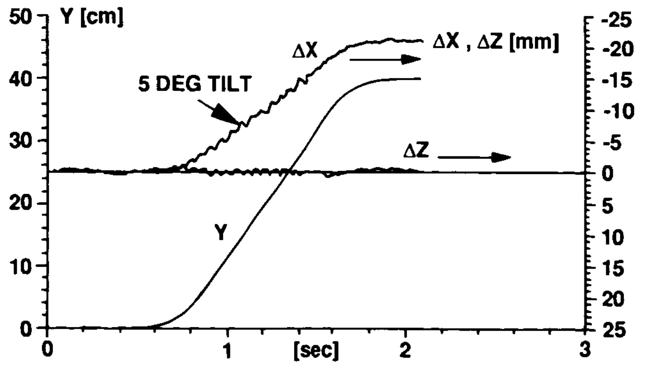


Figure 21. Horizontal (Y) Straight Line Trajectory With Constant X-Force Command and With Cartesian Servo. (Task Board Tilted by 5 Degrees Relative to the Nominal Horizontal Straight Line.)

MANIPULATION II

N91-20647

SSSFD MANIPULATOR ENGINEERING USING STATISTICAL EXPERIMENT DESIGN TECHNIQUES

John Barnes
Martin Marietta Space Systems

Abstract

The Satellite Servicer System Flight Demonstration (SSSFD) program is a series of Shuttle flights designed to verify major on-orbit satellite servicing capabilities, such as rendezvous and docking of free-flyers, Orbital Replacement Unit (ORU) exchange, and fluid transfer. A major component of this system is the manipulator system that will perform the ORU exchange. The manipulator must possess adequate toolplate dexterity to maneuver a variety of EVA-type tools into position to interface with ORU fasteners, connectors, latches, and handles on the satellite, and to move workpieces and ORUs through 6 degree-of-freedom (dof) space from the Target Vehicle (TV) to the Support Module (SM) and back.

Typical of study-phase contracts, a premium is placed on budget, time, and other resourçes to perform trade studies and investigations. Experiments requiring major laboratory hardware builds are almost precluded. Therefore, for this study, two cost efficient tools were combined to perform an investigation of robot manipulator design parameters. These tools are graphical computer simulations and Taguchi Design of Experiment methods. Using a high performance graphics platform, an off-the-shelf robot simulation software package, and an experiment designed with Taguchi's approach, the sensitivities of various manipulator kinematic design parameters to performance characteristics are determined with minimal cost.

Taguchi's methods have been applied to many research and manufacturing problems, but seldom to system design issues. To our knowledge, this is the first application of Taguchi's methods to a manipulator design problem.

1.0 Introduction

Since the SSSFD is to be a minimum cost, high reliability program, the contractor is encouraged to utilize hardware elements of the Flight Telerobotic Servicer (FTS) robot with as few modifications as possible. The challenge, then, is to identify potential areas of manipulator modification which will yield the greatest increases in the quality characteristics of importance to the SSSFD mission. This paper documents the definition, conduct, results, and conclusions of our investigation of manipulator design features, using Taguchi methods for experimental design.

2.0 Performance Characteristics

The SSSFD manipulator will eventually be required to service a variety of satellites with a wide range of ORU sizes, fastener types, and access configurations. It, therefore, will likely require at least 6 dofs, and as much dexterity as possible to be adaptable to changing workspace conditions.

The ultimate purpose of our investigation, then, is to identify potential areas of modification of the manipulator kinematic design which would yield the greatest increase in toolplate dexterity. From our experiences simulating robotic tasks for the FTS program, we defined dexterity as the included angle of tool orientation about a point in the workspace where at least 12" of approach length is available along any ray within that angle. This definition relates to the requirement to position an 8" long tool or end effector to within a 4" approach to the fastener, where final alignment can be effected to interface properly. Since the manipulator's dexterous capability varies over the points in the workspace, it was decided that maximizing the average included angle over these four points will provide the best conditioned workspace for general applications. Our quality characteristic, then, was the included angle of orientation of the toolplate averaged over multiple workspace points.

To gather data, we used the Approach Length Ray Graph (ALRG) generation capability of the SILMA, Inc., CimStation robotic simulation software running on a Silicon Graphics, Inc. IRIS 4D workstation. ALRGs [1] graphically depict the approach length and orientation capabilities at points of interest within any chosen plane. In addition, the software creates data files of the search data for analysis.

In an initial investigation, we chose four points within a normal workspace envelope to take data; three of which were at roughly 75% of total manipulator reach and the fourth at a point in the center, but 20" closer to the base. The three points at 75% reach were defined in the center, upper left, and lower right extremes to get full coverage of the workspace. The results of this initial work identified joint travel limits - especially wrist yaw and pitch joints - as the only significant factor affecting dexterity. It was felt, though, that we did not adequately address ranges of total arm length versus wrist length in this investigation. Also, we did not include interactions between factors or the effects of any noise sources in this experiment. Therefore, a second experiment was designed to study the effects of the significant factors from the initial results as well as interactions and a source of process noise. This paper discusses the details of the second investigation.

At this point, we have not identified the values associated with a Loss Function, although it is clear our Loss Function is based on "Larger is Best," since we want to maximize dexterity. We simply have not quantified the losses resulting from a lack of dexterous capability, although that may be a future task. Our initial interest is to identify the design areas in which to focus our resources.

3.0 Design Factors

The baseline FTS manipulator, shown in Figure 1, is a 7R (7-dof, revolute jointed) robot with six actively controlled joints. Its degrees of freedom are (1) indexable shoulder roll, (2) shoulder yaw, (3) shoulder pitch, (4) elbow pitch, (5) wrist pitch, (6) wrist yaw, (7) wrist roll. The shoulder pitch is offset 9" from the shoulder yaw axis. The baseline upper and forearm link lengths are 18" each. The wrist pitch-to-yaw and yaw-to-tool plate link lengths are 4" and 10.25" respectively. The shoulder roll joint can be rotated through 180° to effect an "elbow up," "elbow out," or "elbow down" orientation, or any angle in between. In theory, the teleoperator will place the shoulder roll joint at an optimum angle for performing a given task prior to initiating that task motion.

Baseline Link/Joint Parameters
Shoulder Roll Joint Limits = +0°/-180°
Shoulder Roll/Yaw Offset = 7"
Shoulder Yaw Joint Limits = +90°/-225°
Shoulder Yaw/Pitch Offset = 9"
Shoulder Pitch Joint Limits = +120°/-90°
Upper Link Length = 21.8"
Elbow Joint Limits = +0°/-180°
Lower Link Length = 21.8"
Wrist Pitch Joint Limits = ±90°
Wrist Pitch/Yaw Link Length = 4"
Wrist Yaw Joint Limits = ±90°
Wrist Roll Joint Limits = ±180°
Wrist Yaw/Toolplate Length = 10.25"

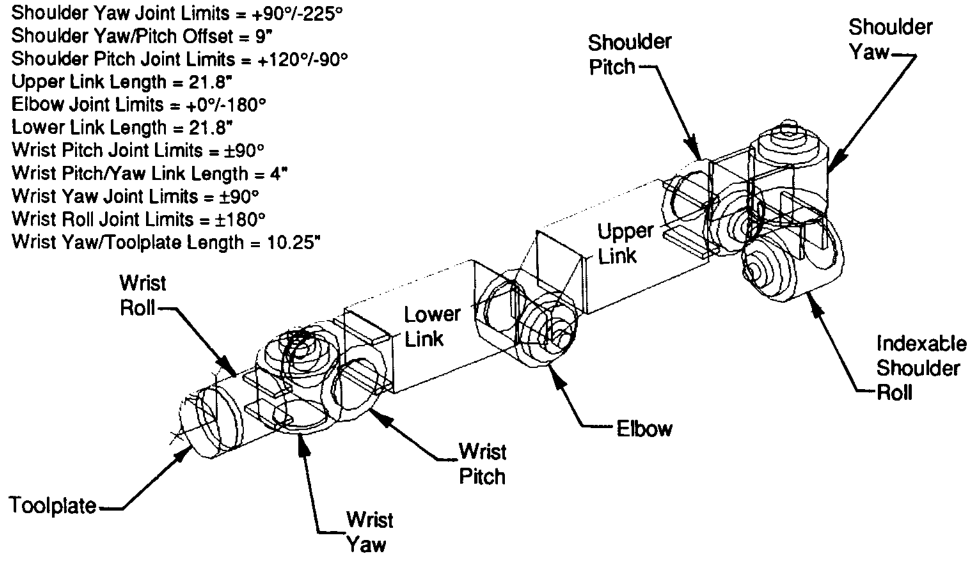


Figure 1 Baseline FTS Manipulator Design

For the experiment, we chose to leave the shoulder link lengths and offsets, and the ordering of the first four joints as they are in the FTS baseline, since they basically perform the function of positioning the wrist and end effector, and don't have a major effect on dexterity. Based on lessons learned from previous task simulations, we defined four design factors with potential to impact dexterity. Three of these concern the wrist design, and the fourth addresses the relative link lengths. The four design factors used were: (1) wrist (or hand) length, measured as the distance from the most proximal joint axis to the toolplate, (2) shoulder pitch and wrist pitch and yaw joint limits, (3) wrist joint configurations, and (4) ratio of arm positioning link lengths to orienting link lengths (shoulder+upper+forearm length to wrist length).

A source of "noise" that could have an impact on available dexterity is the shoulder roll joint position. If we assume that the operator may have less than perfect apriory information on the optimum shoulder roll position for that task, then its position relative to the optimum will vary, and have an effect on dexterity. Also, the shoulder roll joint position may be selected to satisfy requirements other than dexterous capability, such as force transmission or obstacle avoidance. We, therefore, selected three positions for the shoulder roll joint noise variable: (1) 0° (elbow up), (2) 5° elbow outward, and (3) 15° elbow outward.

4.0 Experiment Design

In an initial investigation of an issue, it is recommended [2] to evaluate many design factors at only a few levels. Then, with the results of the first experiment, perform another experiment on the few significant factors at more levels. Since this was our first investigation of FTS manipulator modification effects, we chose to use the four design factors described in Section 3.0 at only two levels, along with the shoulder roll joint "noise" factor in an outer array. These four design factors at their two level variations are shown in Figures 2-5.

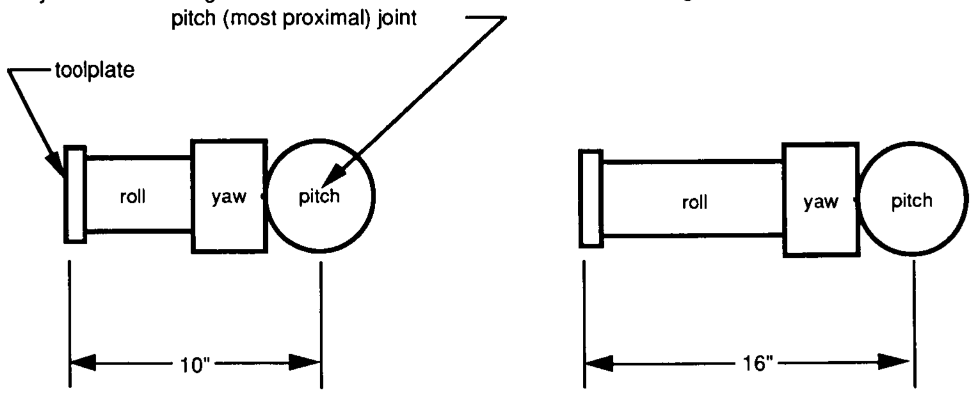


Figure 2 Wrist Length Factor at 10" and 16" Levels

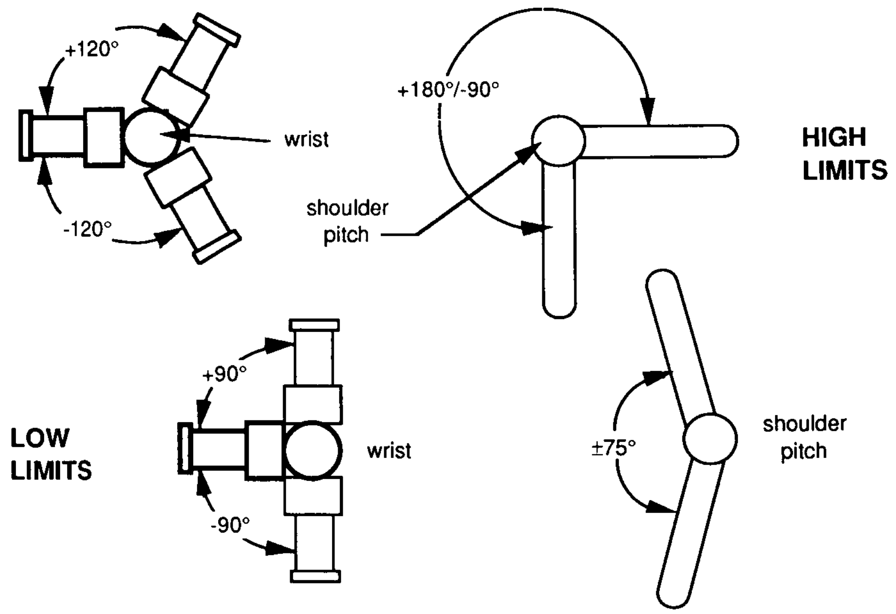


Figure 3 Joint Limits Factor at High and Low Levels

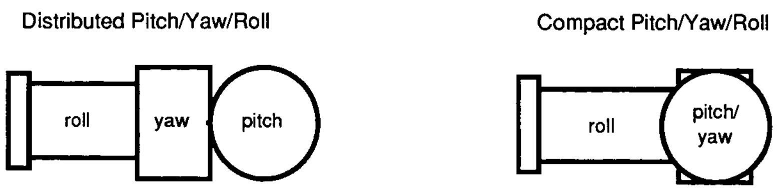


Figure 4 Wrist Joint Configuration Factor at Distributed and Compact Levels

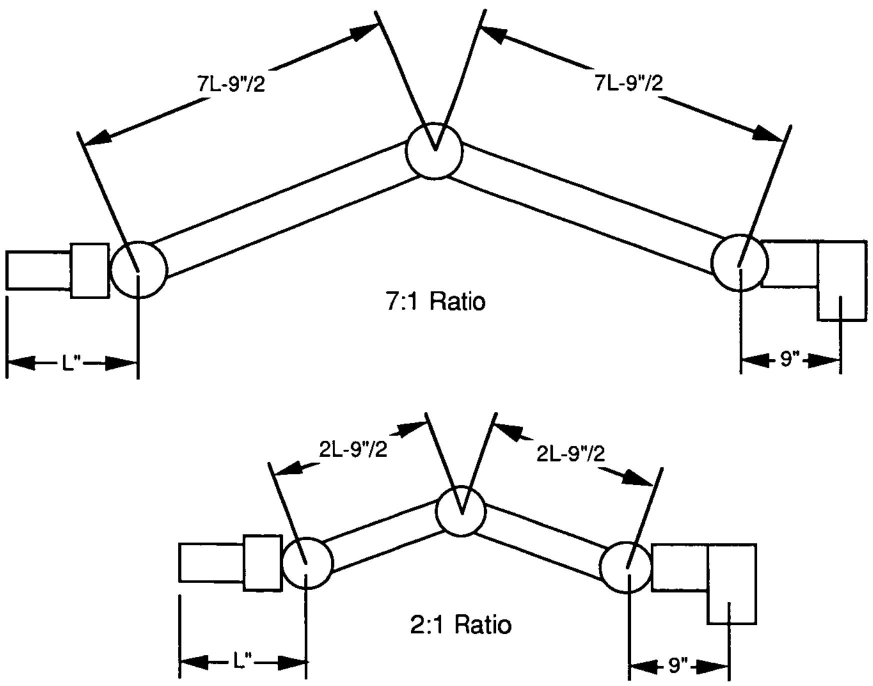


Figure 5 Positioning/Orienting Link Ratio Factor at 7:1 and 2:1 Levels

To fully analyze all possible combinations of these factors at two levels (full factorial experiment) would require creating and collecting data on $2^7 = 128$ different arm configurations. A much more efficient approach developed by Dr. Genichi Taguchi [3] uses orthogonal arrays (or Hadamard's matrices) to define fractional factorial experiments (FFE's) which produce results sufficiently close to those of a full factorial experiment with greatly reduced effort and cost.

For our experiment, we chose to use the $L_8(2^7)$ orthogonal array with an exterior noise array consisting of our three shoulder roll joint positions. This design also allows us to investigate the significance of interactions between some of the factors, specifically between wrist length and joint limits, wrist length and wrist joint configuration, and joint limits and wrist joint configuration. The specific manipulator configurations used in the experiment trials are given in Table 1.

	wrist	joint	wr	link	shoulder	roll	position
trial #	length	limits	design	ratio	0°	5°	15°
1	10"	high	compact	2:1			
2	10"	high	distrib	7:1			
3	10"	low	compact	7:1			
4	10"	low	distrib	2:1			
5	16"	high	compact	7:1			
6	16"	high	distrib	2:1			
7	16"	low	compact	2:1			
8	16"	low	distrib	7:1			

Table 1 Manipulator Experimental Configurations

These manipulator trial configurations were then modeled in the CimStation simulation environment on a Silicon Graphics, Inc. (SGI) 4D/80GT IRIS graphics workstation. The CimStation package contains an application which, when given a manipulator kinematic description, will search for a closed-form inverse kinematic solution. Using this feature, we were able to find closed-form solutions for each arm, which enhanced the validity of our data by guaranteeing all feasible solutions were found for each toolplate position/orientation. Figure 6 shows an example of the CimStation work cell used for each experimental run. The individual manipulator models were installed into the cell and commanded to move their toolplate frame to the appropriate workspace frames, where the ALRG data was generated.

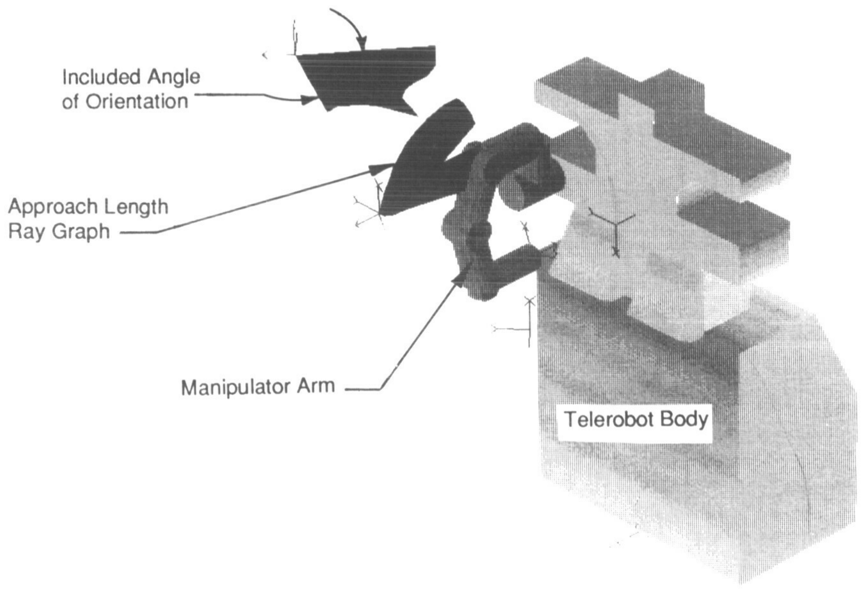


Figure 6 CimStation Workcell for Data Generation

5.0 Experimental Results

Preliminary investigations determined that the dexterity data results are insensitive to the plane (vertical or horizontal) in which the ALRGs are generated. Therefore, data for this experiment were generated in only the vertical plane. The raw average angle data for each of the eight trials is given in Table 2.

	Shoulder	Roll	Position
Trial Number	0°	5°	15°
1	72°	104.8°	102.2°
2	213.8°	213°	191°
3	108.6°	110.8°	111.8°
4	23.6°	31.2°	24.6°
5	221.2°	220.2°	199°
6	116.2°	98.6°	98.2°
7	25.6°	14.8°	13.6°
8	129.8°	126.2°	103.2°

Table 2 Average Dexterous Angle Data for Each Trial

For each of the individual factors and interactions, the raw average angle data are given in Table 3.

noise & factor lvl	wrist length	joint limits	wl x jl	wrist design	wl x wd	jl x wd	link ratio
0° lvl 1	104.50°	155.80°	110.30°	106.85°	106.65°	111.65°	59.35°
0° lvl 2	123.20°	71.90°	117.40°	120.85°	121.05°	116.05°	168.35°
5° lvl 1	114.95°	159.15°	114.70°	112.65°	110.10°	120.60°	62.35°
5° lvl 2	114.95°	70.75°	115.20°	117.25°	119.80°	109.30°	167.55°
15° lvl 1	107.40°	147.60°	102.50°	106.65°	103.85°	107.25°	59.65°
15° lvl 2	103.50°	63.30°	108.40°	104.25°	107.05°	103.65°	151.25°

Table 3 Average Dexterous Angle Data for Each Factor and Interaction

To visualize the results in Table 3, the average response data at each level were plotted at each shoulder roll position, providing a graphic understanding of the effects of level and noise variation on each factor. Figure 7 shows these plots for each design factor and interaction, using a common ordinate axis scale to facilitate direct comparisons of factor responses.

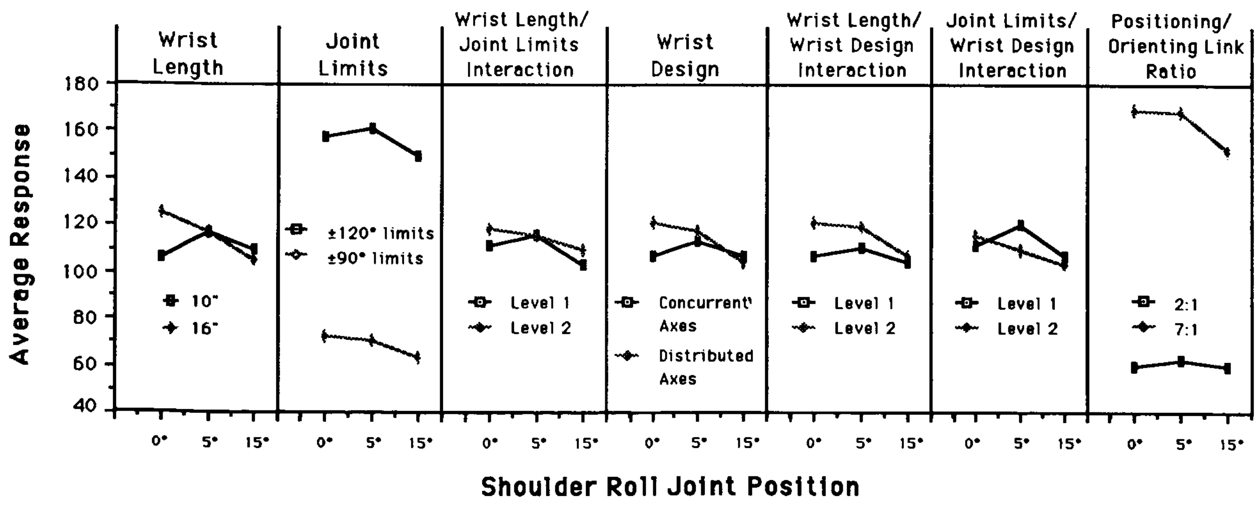


Figure 7 Average Response Data for Each Factor & Interaction

These plots indicate that the only factors or interactions which respond significantly to the changing levels are the Joint Limits and Positioning/Orienting Link Length Ratio factors. They also show that these factors do not respond significantly to the noise source within the 0° - 15° shoulder roll range. The next step was analysis of the data variance to quantify and verify these graphic results.

6.0 Analysis of Variance (ANOVA)

To quantify the sensitivity of the quality characteristic (dexterity) to each design factor and interaction, the contribution of each factor/interaction to the data variance is calculated and compared to the total data variance. This is effectively done by calculating the sums-of-squares (SS) of each factor/interaction and comparing them to the total SS. The total SS about the mean for our data is calculated as follows:

$$SS_T = \Sigma\Sigma y^2 - (\Sigma\Sigma y)^2/N$$

where y = average response for each trial
N = total number of data

Since we are analyzing data for three noise (shoulder roll angle Ø) levels, there are three SS_T values to be calculated:

For Ø = 0°, SS_T = 143180.64 - (829556.64)/8 = **39486.06**

For Ø = 5°, SS_T = 143957.6 - (845664.16)/8 = **38249.58**

For Ø = 15°, SS_T = 120109.68 - (711660.96)/8 = **31152.06**

To determine the significance of the data variation for each design factor and interaction, the individual SS's and their percent contribution to the total variation were calculated. The calculations for the individual factor/interaction SS's are as follows:

$$SS_{F_i} = [(F1)^2/n1 + (F2)^2/n2 + ... + (Fi)^2/n_i] - (\Sigma\Sigma y)^2/N$$

where F_i is the design factor/interaction, and
n_i are the number of data points associated with each factor/interaction

Once again, to ascertain the effects of the noise factor these values were calculated for each noise level.

For Ø = 0°:
SS_{wl} = (43681) + (60712) - (829556.64/8) = **699.38**
SS_{jl} = (97094.56) + (20678.44) - (829556.64/8) = **14078.42**
$SS_{wl\ x\ jl}$ = (48664.36) + (55131.04) - (829556.64/8) = **100.82**
SS_{wjd} = (45667.69) + (58418.89) - (829556.64/8) = **392**
$SS_{wl\ x\ wjd}$ = (45496.89) + (58612.41) - (829556.64/8) = **414.7**
$SS_{jl\ x\ wjd}$ = (49862.89) + (53870.41) - (829556.64/8) = **38.72**
SS_{lr} = (14089.69) + (113366.89) - (829556.64/8) = **23762**

For Ø = 5°:
SS_{wl} = (52854.01) + (52854.01) - (829556.64/8) = **0.0**
SS_{jl} = (101314.89) + (20022.25) - (829556.64/8) = **15629.12**
$SS_{wl\ x\ jl}$ = (52624.36) + (53084.16) - (829556.64/8) = **0.5**
SS_{wjd} = (50760.09) + (54990.25) - (829556.64/8) = **42.32**
$SS_{wl\ x\ wjd}$ = (48488.04) + (57408.16) - (829556.64/8) = **188.2**
$SS_{jl\ x\ wjd}$ = (58177.44) + (47785.96) - (829556.64/8) = **255.38**
SS_{lr} = (15550.09) + (112292.01) - (829556.64/8) = **22134.08**

For Ø = 15°:
SS_{wl} = (46139.04) + (42849) - (829556.64/8) = **30.42**
SS_{jl} = (87143.04) + (16027.56) - (829556.64/8) = **14212.98**
$SS_{wl\ x\ jl}$ = (42025) + (47002.24) - (829556.64/8) = **69.62**
SS_{wjd} = (45496.89) + (43472.25) - (829556.64/8) = **11.52**
$SS_{wl\ x\ wjd}$ = (43139.29) + (45838.81) - (829556.64/8) = **20.48**
$SS_{jl\ x\ wjd}$ = (46010.25) + (42973.29) - (829556.64/8) = **25.92**
SS_{lr} = (14232.49) + (91506.25) - (829556.64/8) = **16781.12**

Since these were purely kinematic simulations with no sources of error, the trials were not repeated, hence there is no repetition error involved. Therefore, the variations given for the factors and interactions represent the total data variation present. This also means that all the information from the experimental results is accounted for in the factors and interactions.

The percent contribution of each factor/interaction to the total data variance, given in Table 4, is calculated by simply ratioing the factor/interaction SS's with the SS_T.

Noise Level	Wrist Length	**Joint Limits**	wl x jl interact	Wrist Design	wl x wd interact	jl x wjd interact	**P/O Ratio**
Ø = 0°	1.77	**35.65**	0.26	0.99	1.05	0.10	**60.18**
Ø = 5°	0.00	**40.86**	0.00	0.11	0.49	0.67	**57.87**
Ø = 15°	0.10	**45.62**	0.22	0.04	0.07	0.08	**53.87**

Table 4 Percent Contribution of Factors & Interactions to Total Data Variance

The Table 4 results show quantitatively what the raw experimental results of Section 5.0 show graphically - that the only factors or interactions with significant contribution to data variance are Wrist Joint Limits and Positioning/Orienting Link Length Ratio. These composite percentage results are also shown graphically in bar-chart form in Figure 8.

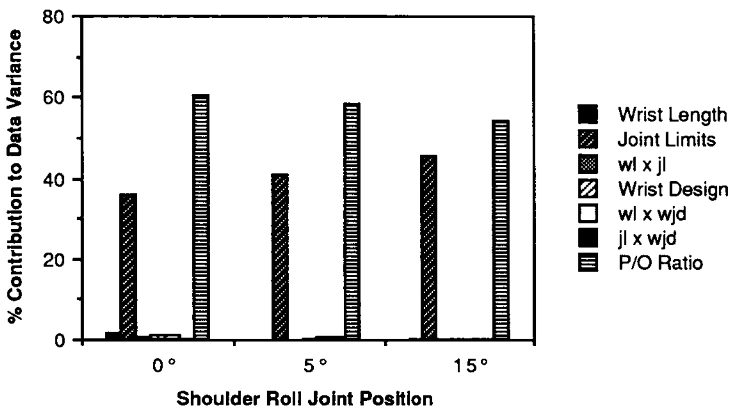


Figure 8 Factor Contributions to Total Data Variance for each Noise Level Value

8.0 Confirmation Trial

A confirmation trial was run with a manipulator configuration that uses the best joint limits and positioning/orienting link ratio, but leaves the other factors at the current FTS manipulator baseline design. In other words, this configuration uses shoulder pitch joint limits of +180°/-90°, a 16" distributed actuator wrist with ±120° pitch and yaw joint limits, and has upper arm and forearm link lengths of 51.5". Since the noise source had an insignificant effect on data variation relative to the design factor levels, this trial was performed in the 0° shoulder roll position only, using the same workspace points as configuration #5.

The results of this trial indicate an average orientation angle of 220.8°, which is insignificantly lower than the best scoring configuration - configuration #5 (concurrent axis wrist) - as would be expected.

8.0 Summary and Conclusions

These experimental results point to two potential manipulator modifications which would yield significant gains in dexterous capability for the SSSFD program: (1) increasing the FTS manipulator shoulder pitch joint limits to +180°/-90° and wrist pitch and yaw limits from ±90° to ±120°, and (2) increasing the positioning-to-orienting link length ratio as much as feasible. They also indicate that - within a range allowing reach to the workspace points - the shoulder roll joint position has an insignificant effect on dexterity.

Of these two factors, the link ratio has a slightly higher performance yield if taken from 2:1 to the 7:1 level, however, it also involves significant impacts to weight, stiffness, power consumption, thermal control, and stowed volume, especially if tip force and control input bandwidth performance is to be maintained. In addition, a preliminary experiment which varied the link ratio from 5:1 to 9:1 showed insignificant impact to dexterity, indicating a function which rises rapidly above 2:1 and levels off somewhere prior to 5:1. This factor should be investigated in more levels between 2:1 and 5:1.

The joint travel limit factor - although not a trivial design issue - is probably the best candidate for modification to improve dexterity.

REFERENCES

1.Gupta, K. C., "On the Nature of Robot Workspaces," International Journal of Robotics Research, Vol. V, No. 2, 1986

2.Ross, Phillip J., *Taguchi Techniques for Quality Engineering*, McGraw-Hill Book Company, 1988

3.Terninko, John, *Introduction to Quality Engineering Using the Philosophy of Dr. Genichi Taguchi*, Responsible Management, Durham, N. H.,1989

N91-20648

IMPACT OF INERTIA, FRICTION AND BACKLASH UPON FORCE CONTROL IN TELEMANIPULATION

Neil A. Duffie, Associate Professor
John J. Zik, Assistant Researcher

Steven F. Wiker, Assistant Professor
Karen L. Gale, Assistant Researcher

Wisconsin Center for Space Automation and Robotics
University of Wisconsin-Madison
1357 University Ave., Madison, WI 53715

ABSTRACT

The mechanical behavior of master controllers of telemanipulators has been a major concern of both designers and implementors of telerobotic systems. In general, the literature recommends that we construct telemanipulator systems that minimize inertia, friction, and backlash in an effort to improve telemanipulative performance. For the most part, these recommendations are founded upon theoretical analysis or simply intuition. Although we do not challenge the recommendations on their merit, we were interested in measuring the material consequences of building and fielding telemanipulators that possess less than ideal mechanical behaviors. Experiments are described in this paper in which forces in a mechanical system with human input are evaluated as a function of mechanical characteristics such as inertia, friction and backlash. Results indicate that the ability of the human to maintain gripping forces was relatively unaffected by dynamic characteristics in the range studied, suggesting that telemanipulator design in this range should be based on task-level force control requirements rather than human factors.

INTRODUCTION

Designers of telerobotic systems are often faced with important trade-offs concerning the mechanical characteristics of the manipulator mechanisms and their impact upon the performance capability of the human operator. For example, a designer can use direct-drive actuators to substantially reduce backlash and friction, but to do so requires the use of larger actuators with greater inertial characteristics. Smaller geared-drives can be used but not without encountering higher levels of backlash and friction. It is possible to reduce backlash in geared drives, but not without increasing friction to some degree. Finally, most designers would prefer to minimize or compensate for friction in telemanipulators, caused by gearing, cables, etc. Unfortunately, friction is both difficult to eliminate and difficult to model and predict accurately, making friction compensation in control systems difficult even though complex compensation algorithms can be implemented in computer software.

Knowing the relative consequences and interrelationships among mechanical properties of telemanipulators, in terms of human controller performance, provides:

a) opportunities for confident and strategic selection of telemanipulator system components with tolerable levels of inertia friction, and backlash; and

b) greater opportunity for diversity and competition among master-slave controller designs (e.g.,degrees-of-freedom, actuators, etc.).

A dominant performance requirement for effective telemanipulation is timely and accurate operator detection and control of remote grasp forces. This is a particular problem when teleoperating manipulators in remote environs where the opportunities for unexpected disturbances in remote grasp are high. Disturbances in grasp can result from sudden forces applied by the object within the gripper, or by the manipulator, which result in rapid movement and/or changes in the forces between the object and the manipulator. The net effect can be either complete loss of contact with the object, or object slippage and realignment within the gripper. This certainly lengthens the job if the operator must regrasp and reorient either the object or the manipulator arm. Thus, high-performance, or at least operationally acceptable, manipulation depends on the interrelationships between quality of force feedback information and manipulator dynamics.

The objective of this study was to determine whether realistic variations in inertia, friction, and backlash or deadspace, produced material challenges to the human operator's capacity to control grasp force in the face of an unexpected disturbance.

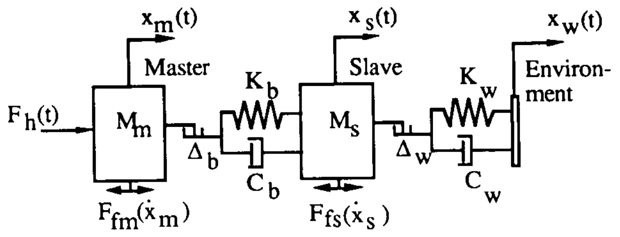
METHODS AND MATERIALS

Subjects

Eight males and one female ranging in age from 20 to 36 years participated in the experiment. All subjects reported and appeared to be in good health with no history of neuromuscular disorders. Participation in the experiment was on an informed consent, voluntary, and paid basis.

Apparatus

A mechanical system model of a one degree-of-freedom, bilateral, master/slave system with the slave in contact with the work environment is shown in Figure 1.



Nomenclature:

M_m,M_s :	Inertia of the master and slave
K_b,C_b:	Stiffness and damping in the master/slave system
Δ_b,Δ_w:	Backlash in the master/slave system and work environment
K_w,C_w:	Stiffness and damping between the slave manipulator and the work environment
$F_h(t)$:	Force applied by the human
x_m,x_s,x_w:	Position of the master, slave and work environment, respectively
F_{fm}, F_{fs}:	Friction force on master and slave, respectively

Figure 1: Master/slave system model

The model is non-linear due to the incorporation of friction and backlash. In an actual master/slave system with a number of power transmitting components there will be a series of masses and associated backlashes. However, these can generally be lumped into equivalent masses, backlash, etc. [1]. Friction can take various forms such as dry, fluid, etc., and is always a resistive force that is dissipative and has a retarding effect on the motion of the system [2].

For the purposes of our experiment, the bilateral system described in Figure 1 was simplified to that shown in Figure 2. The simplified system model can represent a case in which the interface between the slave device and the work environment (the object being manipulated) is relatively rigid (K_w is large and Δ_w is small) and the master/slave system is relatively compliant, or a case in which the interface between the slave device and the work environment is relatively compliant and the master/slave system is relatively rigid (K_b and C_b are large, and Δ_b is small). In the former case, the slave device can be considered to be coupled rigidly to the environment, with dynamic characteristics lumped between the master and slave. In the latter case, the master can be considered to be coupled rigidly to the slave, with dynamic characteristics lumped between the slave and the work environment. In all cases, the force applied by the human in the simplified model, $F_h(t)$, corresponds to the force applied by the human to the master device in Figure 1.

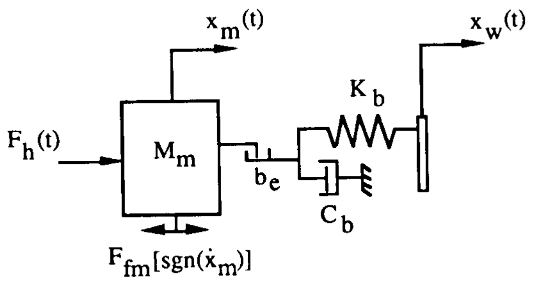


Figure 2. Simplified system model implemented in experiments

The apparatus system used in the experiments employed direct drive actuators, and hence had no intrinsic backlash [3]. Mechanical friction was also minimized in the system by using brushless motors and a precision linear slide. The system used was nearly an ideal, linear second-order system with a maximum positioning natural frequency of 30 Hz.

A strain gauge force sensor was used to measure forces exerted by the subjects. A high-resolution encoder was employed for position feedback and velocity estimation. A microcomputer controlled the position of the actuator and recorded: a) position commands; b) actual position; c) strain-gauge voltages; and d) computer clock time. A schematic of the experimental configuration is shown in Figure 3.

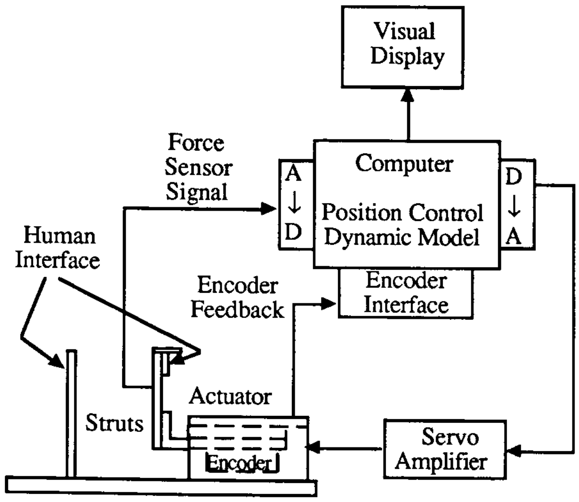


Figure 3. Experimental apparatus

Stiffness, damping, inertia, friction and backlash characteristics perceived by the human subject were programmed and controlled by system software. Apparent backlash was implemented by providing a dead band between commanded position and the actual position. Apparent friction was produced using an algorithm with velocity of the mass and the forces on the mass as inputs. Impedance control techniques were employed for implementing the desired stiffness, damping and mass parameters [4].

Procedures

Following an initial period of practice with the task, subjects were asked to grasp the struts on the apparatus (one fixed and one connected to the actuator) with the thumb and index finger. The subjects were instructed to squeeze, using a pulp-pinch grasp, until they achieved a 5 N force. The level of force was indicated by movement of a computer screen cursor to a visual target. The subjects held the force until they were confident that they could recognize and return to the 5 N force if a sudden loss of force was experienced

Following the subject's signal to begin, the computer would move the position of the apparent work environment, $x_w(t)$, 6 mm in the direction away from the subject's index finger. This step-displacement, which was produced at a random interval between 2 and 7 s after the subject signalled the start of the trial, decreased the force acting against the subject's finger. The subject's goal was to maintain a constant 5 N force at all times regardless of a positional disturbance. The computer began to record at a 166.7 Hz rate at 2 s before the 6 mm step occurred and recorded for another 6 seconds following the step.

Experimental Design and Analysis

To simplify the experiment, the stiffness and damping parameters were held constant (K = 3.7186 N/mm, C = 0.10677 N-s/mm). Earlier testing showed that variations in these parameters did not have a material affect upon grasp force control within the limits of the independent variables studied in this experiment. Static frcition was equal to dynamic (coulomb) friction in the tests. Each subject repeated a trial 5 times under each of 27 combinations of three-levels of

Table I: System configurations used in experiments

Config. Number	Mass (Kg)	Friction (Newtons)	Backlash (mm)	Nat. Freq (hz)	Damping Ratio
1	0.6813	0.0	0.00	11.76	1.06
2	0.6813	0.0	0.25	11.76	1.06
3	0.6813	0.0	0.50	11.76	1.06
4	0.6813	1.5	0.00	11.76	1.06
5	0.6813	1.5	0.25	11.76	1.06
6	0.6813	1.5	0.50	11.76	1.06
7	0.6813	3.0	0.00	11.76	1.06
8	0.6813	3.0	0.25	11.76	1.06
9	0.6813	3.0	0.50	11.76	1.06
10	1.3626	0.0	0.00	8.31	0.75
11	1.3626	0.0	0.25	8.31	0.75
12	1.3626	0.0	0.50	8.31	0.75
13	1.3626	1.5	0.00	8.31	0.75
14	1.3626	1.5	0.25	8.31	0.75
15	1.3626	1.5	0.50	8.31	0.75
16	1.3626	3.0	0.00	8.31	0.75
17	1.3626	3.0	0.25	8.31	0.75
18	1.3626	3.0	0.50	8.31	0.75
19	2.7252	0.0	0.00	5.88	0.53
20	2.7252	0.0	0.25	5.88	0.53
21	2.7252	0.0	0.50	5.88	0.53
22	2.7252	1.5	0.00	5.88	0.53
23	2.7252	1.5	0.25	5.88	0.53
24	2.7252	1.5	0.50	5.88	0.53
25	2.7252	3.0	0.00	5.88	0.53
26	2.7252	3.0	0.25	5.88	0.53
27	2.7252	3.0	0.50	5.88	0.53

backlash, friction, and inertia shown in Table I. Each subject experienced all combinations of the experimental conditions in a random order.

The mean grasp force time history following the step disturbance of 5 trials served as the performance metric under each of the test conditions. Grasp control performance was characterized using each of the following metrics:

a) magnitude of force loss following the step disturbance;

b) time needed by the subject to return to 4.5 N or 90 percent of the initial grasp force (referred to as force recovery period); and

c) magnitude of grasp force following grasp recovery (mean force recorded during the last 2 s of the trial).

The above metrics were examined using randomized-block ANOVA to determine whether or not inertia, friction, backlash, or any two and three-way interactions were significant. All tests were conducted fixing Type I and Type II errors at p=0.05 and p=0.10 respectively.

RESULTS

Magnitude of Force Loss

As the apparent work environment stepped away from the grasp of the subject, pinch grasp force declined. The ideal response would show no change in force level or zero force loss.

As shown in Figure 4, the average magnitude of this decline was largely unaffected by the different combinations of mass, friction, and backlash experienced. Increasing the level of backlash did produce only slightly greater losses in force ($F = 8.13$; $df = 2,16$; $p = 0.0037$); however, as shown in the figure the effect was not material in nature. All remaining effects, as well as their interactions, were not statistically significant ($p \geq .05$; Power $\geq .90$).

Time Period Needed for Recovery of Grasp Force

The period of time needed for the subject to recover 90 percent of the original grasp force following the step should be kept as small as possible. The analyses indicated that following the step disturbance, recovery times were essentially the same regardless of inertial, friction, and backlash characteristics examined.

Magnitude of Grasp Force Following Grasp Recovery

Ideally, the subject should recover from the loss of force following the disturbance, and return grasp force back to the initial levels. The ANOVA results revealed that differences in mass, friction, and backlash had no effect upon the level of force established following recovery from the disturbance. However, subjects almost always produced greater than 5 N of grasp force upon reestablishing their perceived grasp force goal.

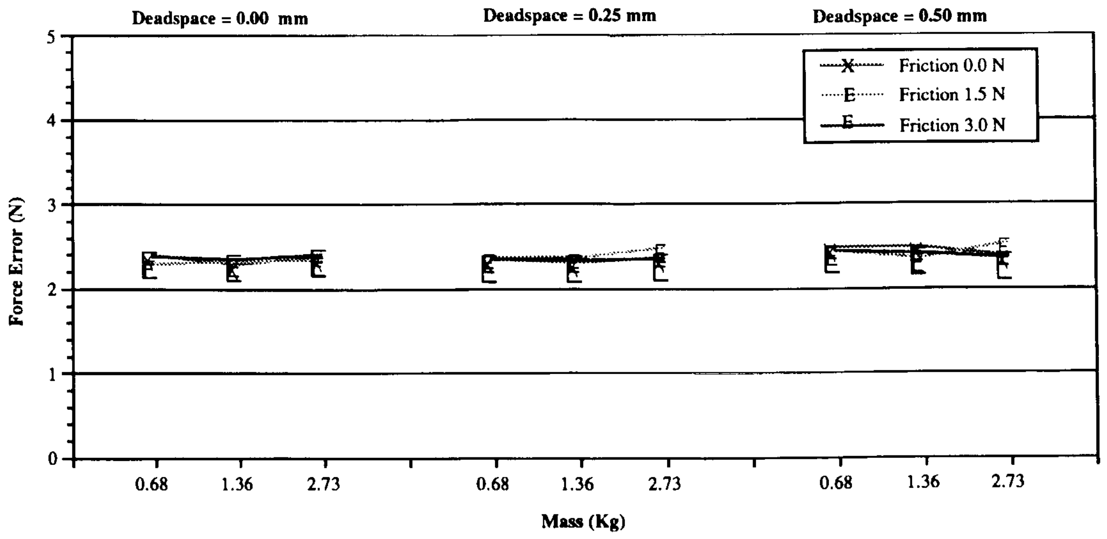


Figure 4. Magnitude of force loss

Interrelationships Among Force Error, Force Recovery Period, and Shift in Force Baseline.

The Pearson-product moment correlations showed no meaningful relationships existed between measures of subject force control performance and variations in levels of mass, friction, or backlash.

DISCUSSION

Plots of force time histories, such as those in Figure 5, showed that subjects began to actively recover control of force after about 140 ms, and that their restoration behavior was similar to an over-damped second-order force response with a dominant time constant of approximately 280 ms. During the initial 140 ms following the step, the grasp force does not fall to zero.

The hand is actively controlling forces prior to the step. As the actuator moves away from the finger, the active tension set of the extrinsic and intrinsic musculature of the hand is impeded only by the actuator. Thus, the finger initially "follows" the movement, continuing to apply more than half of the original 5 N.

Once the loss of force is detected, the subject actively contracts affected muscles to return to a pre-step level of tension. The rate of return, or force recovery period, is largely instruction-dependent. The rate we observed was established by the subject's need to restore force as quickly and as accurately as possible without overforcing. Thus, the subject attempted to produce a controlled over-damped response without overshooting the perceived level or muscle tension needed to regain the desired grasp force. Had the subjects been instructed to recover grasp force as quickly as possible without concern about overforcing, they would have produced much shorter force recovery periods, would have overshot the grasp force goal, and then reduced grasp force to the perceived goal. In this experiment, the subjects tended to return to stable force levels that were slightly greater than the original 5 N.

Deadspace was experienced during the initial motion of the actuator. This space was traversed passively by the finger as it "followed" the displaced actuator. The finger had preloaded the actuator again prior to the initiation of active force control by the finger. Once the finger had passed through the deadspace, backlash no longer existed. This left the subject facing only mass and friction effects when returning forces to initial levels.

Static and dynamic friction forces, from the perceptual perspective of the subject, appear to be lumped with the inertial or mass effects. Thus, the operator perceives the force, whether due to inertial or friction effects, as an equivalent cue.

The plots in Figure 5 also reveal that the human subjects applied restoring forces in a very consistent manner even though the apparent reaction force was a result of different combinations of forces (friction, inertial, spring and damper). This suggests that the human subjects may treat all reaction forces similarly. However, as can be seen in Figure 6, there are differences in the work environment force response for various combinations of dynamic characteristics. These are due the mechanical properties, and the differences appear to be independent of human input.

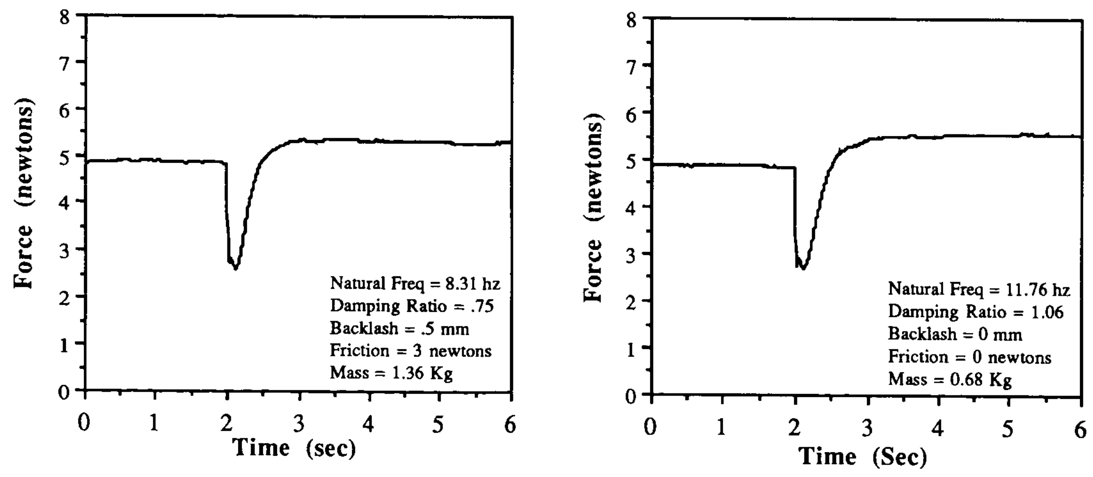
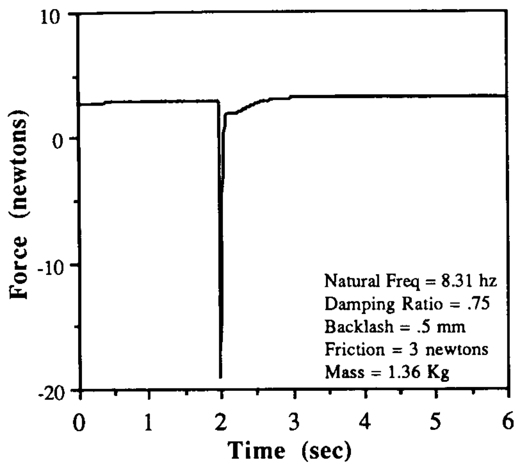


Figure 5. History of force applied by human subject



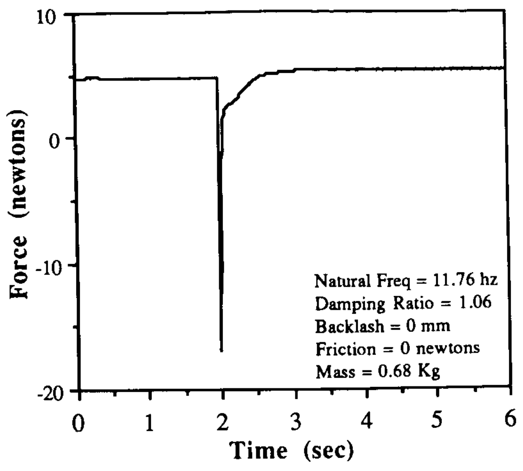


Figure 6. History of work environment force

CONCLUSIONS

From our findings it appears the operator perceives forces, whether due to inertial or friction effects, as equivalent cues. This indicates that master/slave manipulator dynamics may be more important than human force control characteristics in high-performance telemanipulation in the force domain. It appears that the human operator is tolerant of reasonable levels of master-controller mass, friction, and backlash characteristics when compelled to maintain grasp forces within desired operating ranges. It is not clear whether these conclusions would hold for higher levels of mass, friction, or backlash that those addressed in our experiment. Preliminary studies indicate that if there is a significant difference between the static and dynamic coefficients of friction, then acceptable force control performance becomes more difficult to achieve.

ACKNOWLEDGMENTS

This work was supported by the Wisconsin Center for Space Automation and Robotics in part by NASA under grant NAG-W975 and by a consortium of industrial sponsors.

REFERENCES

1. Goodman, T., "Dynamic Effects of Backlash", Machine Design, May 23, 1963 pp 150-157.

2. Tabor, D., "Friction - The Present State of Our Understanding", Journal of Lubrication Technology, Vol. 103, April, 1981, pp 169-179.

3. Duffie, N.A., Wiker, S. F. and Zik, J. J., "Test Bed Experiments for Various Telerobotic system Characteristics and Configurations", Proceedings of the Annual SOAR Conference, Houston, TX, July, 1989.

4. Hogan, N., "Impedance Control of Contact Tasks Using Force Feedback", Undersea Teleoperators and Intelligent Autonomous Vehicles, Ed. Doelling, N., Harding, E., MIT Sea Grant College Program, January, 1987.

5. Wiker, S. F., Hershkowtiz, E., Zik, J. J., "Teleoperator Comfort and Psychometric Stability: Criteria for Limiting Master-controller Forces of Operation and Feedback During Telemanipulation", Proceeding of NASA's Conference on Space Telerobotics, Jet Propulsion Laboratory, Pasadena, CA, January, 1989.

N91-20649

Adaptive Control of Space Based Robot Manipulators

Michael W. Walker and Liang-Boon Wee

Artificial Intelligence Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

Abstract

For space based robots in which the base is free to move, motion planning and control is complicated by uncertainties in the inertial properties of the manipulator and its load. This paper presents a new adaptive control method for space based robots which achieves globally stable trajectory tracking in the presence of uncertainties in the inertial parameters of the system.

The paper begins with a partitioning of the fifteen degree of freedom system dynamics into two components: a nine degree of freedom invertible portion and a six degree of freedom noninvertible portion. The controller is then designed to achieve trajectory tracking of the invertible portion of the system. This portion of the system consist of the manipulator joint positions and the orientation of the base. The motion of the noninvertible portion is bounded, but unpredictable. This portion of the system consist of the position of the robot's base and the position of the reaction wheels.

1 Introduction

In recent years the control of space based manipulators has received increased attention. The main difference between space based robots and their terrestrial counterpart is the dynamic coupling between the manipulator and its floating base. This results in a similar, but uniquely different form for the kinematic and dynamic equations of motion.

Several researchers have focused on the forward and inverse kinematics problem [1, 2, 3, 4]. One of the interesting parts of these results is the formulation of the *dynamic* Jacobian matrix. It is now recognized that singularities may occur in the transformation from end-effector velocities to joint velocites which are at different locations than the normal kinematic singularity points. Another interesting result is the concept of the virtual manipulator, [2]. If the mass properties of each link are known, then it can be shown that a virtual manipulator can be obtained for use in the inverse kinematics problem. The advantage of the virtual manipulator is that algorithms developed for terrestrial based manipulators can be applied.

The dynamics of multibody space based systems has been researched for many years [5, 6, 7]. In many ways the control of space based robots is similar to the problems traditionally faced in satellite control. The main difference is the articulated nature of the robot. Free floating space based robot control has only recently gained attention [8, 9].

In both the kinematics problem and the control problems previous researchers have assumed either the mass properties of the system are completely known or the momentum of the system is zero. This paper presents a control method in which neither of the assumptions are made.

We begin this paper with a description of the system considered and the formulation of the dynamic equations of motion. The fifteen degree of freedom system dynamics are partitioned into two components: a nine degree of freedom invertible portion and a six degree of freedom noninvertible portion. The invertable portion of the system consist of the manipulator joint positions and the orientation of the base. The motion of the noninvertible portion is bounded, but unpredictable. This portion of the system consist of the position of the robot's base and and the velocity of the reaction wheels. An adaptive controller is then presented to achieve trajectory tracking of the invertible portion of the system. Finally, a summary of the main results and conclusions of the paper are presented.

2 Equations of Motion

The system we are considering is an n degree of freedom serial link manipulator, with rotational or translational joints, mounted on a base containing three reaction wheels. It is assumed that no external forces are moments are applied to the system. However, no assumptions have been made concerning the initial momentum of the system.

Associated with each link is a right handed Cartesian coordinate system whose position and orientation is fixed with respect to the associated link. This is illustrated for link j in Figure 1. The location of this coordinate frame with respect to an inertial reference frame is denoted by the homogeneous transform $\boldsymbol{T}_j$.

A Floating Referenced Frame is fixed at a specified position on the Base with the same orientation as the inertial reference frame. Its location is denoted by the homogeneous transform $\boldsymbol{T}_0$.

The Base link is numbered 3 and its coordinate frame is located at the same location as the Floating Reference Frame, but at a different orientation. It's location is denoted by the

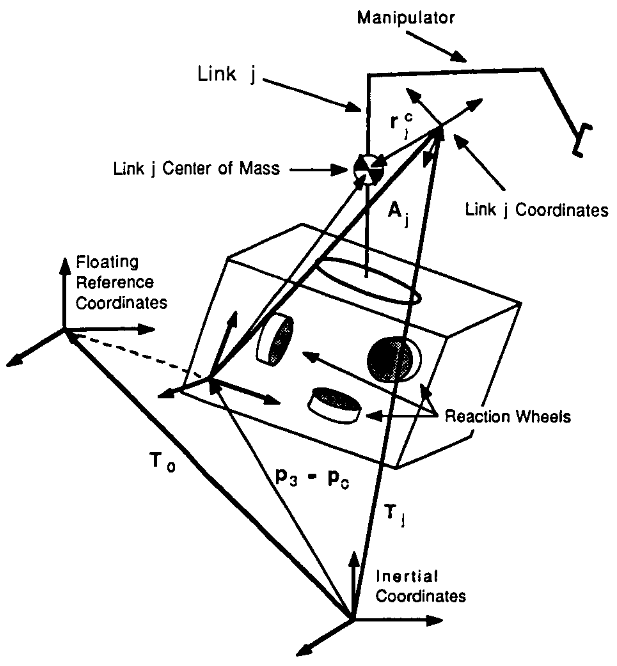


Figure 1: Illustration of Robot

homogeneous transform, T_3. Between the Floating Reference Frame and the Base frame are two fictitious links of zero mass. The three joints between these links represent the relative change in orientation of the Base with respect to the Floating Reference Frame.

The manipulator is attached to the Base and the links are numbered from 4 to $n+3$. In addition three reaction wheels are located inside the Base. The wheels are numbered from $n+4$ to $n+6$.

The configuration of the complete system is a tree structure as illustrated in figure 2 for the case of $n = 6$. Considering the Floating Reference Frame to be the base of the tree, each joint in the system is numbered the same as the immediate descendant in the tree. The position of the $i-th$ joint is denoted by q_i.

The kinetic energy of the system is given by the following equation.

$$K = \sum_{i=3}^{n+6} \frac{1}{2} TR\{\dot{\boldsymbol{T}}_i \boldsymbol{D}_i \dot{\boldsymbol{T}}_i^T\} \tag{1}$$

where $\boldsymbol{D}_i$ is the constant pseudo inertia matrix for link i referred to link i coordinates. [10, 11], and $TR\{\}$ denotes the trace operator.

$$\boldsymbol{D}_i = \int \boldsymbol{r}\boldsymbol{r}^T dm = \begin{bmatrix} \int x^2 dm & \int xydm & \int xzdm & \int xdm \\ \int xydm & \int y^2 dm & \int yzdm & \int ydm \\ \int xzdm & \int yzdm & \int z^2 dm & \int zdm \\ \int xdm & \int ydm & \int zdm & \int dm \end{bmatrix}$$

where the integration is carried out over the entire link, and $\boldsymbol{r} = [x, y, z, 1]^T$ is the position vector of the mass element with respect to link i coordinates.

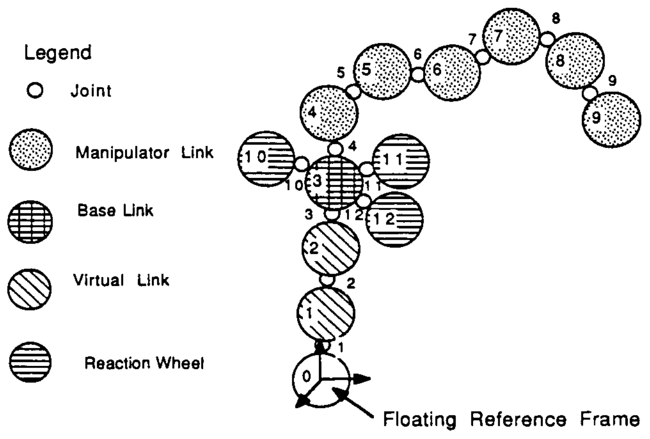


Figure 2: Configuration of Space Based Robot

2.1 Reaction Wheels

The torque delivered to the base of the robot to control its orientation is provided by a set of reaction wheels. The position variables associated with these wheels are cyclic and therefore it considerably simplifies the analysis by writing the kinetic energy and the resulting equations of motion in terms of the generalized momentum, l_j, associated with these wheels.

$$l_j = \frac{\partial K}{\partial \dot{q}_j}$$

The main objective of this section is to write the kinetic energy kinetic energy in terms of these generalized momentum. To this end, we consider the kinetic energy of the $j-th$ reaction wheel,

$$K_j = \frac{1}{2} TR\{\dot{\boldsymbol{T}}_j \boldsymbol{D}_j \dot{\boldsymbol{T}}_j^T\} \tag{2}$$

We will show this can be written in the following form:

$$K_j = \frac{1}{2} TR\{\dot{\boldsymbol{T}}_3 \boldsymbol{E}_j \dot{\boldsymbol{T}}_3^T\} + \frac{1}{2} J_j^{-1} l_j^2$$

where J_j is the moment of inertia of the $j-th$ reaction wheel about it's axis of rotation and $\boldsymbol{E}_j$ is a constant matrix.

We begin with some notation. Let $\boldsymbol{x}$ be an arbitrary 6×1 vector, which has been partitioned into two 3×1 vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}$$

and define the matrix function $\boldsymbol{R}(\boldsymbol{x})$ as

$$\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{a}) & \boldsymbol{b} \\ 000 & 0 \end{bmatrix}$$

where $\boldsymbol{k}()$ is a 3×3 matrix function such that for any two 3×1 vectors $\boldsymbol{a}$ and $\boldsymbol{y}$, $\boldsymbol{k}(\boldsymbol{a})\boldsymbol{y} = \boldsymbol{a} \times \boldsymbol{y}$, where $\times$ denotes the vector cross product.

With this notation, we can write the time derivative of $\boldsymbol{T}_j$ in the form:

$$\dot{\boldsymbol{T}}_j = \boldsymbol{R}(\underline{\boldsymbol{v}}_j)\boldsymbol{T}_j \tag{3}$$

where

$$\underline{\boldsymbol{v}}_j = \underline{\boldsymbol{v}}_0 + \sum_{i=1}^{3} \underline{\boldsymbol{s}}_i \dot{q}_i + \underline{\boldsymbol{s}}_j \dot{q}_j$$

and,

$$\underline{v}_0 = \begin{bmatrix} 0 \\ \dot{\underline{p}}_0 \end{bmatrix}$$

$$p_0 = \begin{bmatrix} \underline{p}_0 \\ 1 \end{bmatrix}$$

The vector $\underline{p}_0$ is the position vector of the Floating Reference Frame and, in general, for any joint k,

$$\underline{s}_k = \begin{cases} \begin{bmatrix} \underline{n}_k \\ \underline{r}_k \times \underline{n}_k \end{bmatrix} & \text{if joint } k \text{ is rotational} \\ \begin{bmatrix} 0 \\ \underline{n}_k \end{bmatrix} & \text{if joint } k \text{ is translational} \end{cases} \tag{4}$$

where $\mathbf{0}$ is the null vector and $\underline{n}_k$ is a unit vector along the axis of rotation if the joint is rotational, or a unit vector in the direction of translation if the joint is translational. The vector $\underline{r}_k$ is a position vector of an arbitrary point on the axis of rotation of the joint if rotational. The vectors $\underline{n}_k$, $\underline{r}_k$, and $\underline{p}_0$ are defined relative to the inertial coordinate frame.

With this notation we can write:

$$K_j = \frac{1}{2} TR\{\boldsymbol{R}(\underline{v}_j)\boldsymbol{T}_j\boldsymbol{D}_j\boldsymbol{T}_j^T\boldsymbol{R}(\underline{v}_j)^T\}$$

The matrix $\boldsymbol{T}_j\boldsymbol{D}_j\boldsymbol{T}_j^T$ is the pseudo inertia matrix of the $j-th$ reaction wheel referred to the inertial coordinate frame. We now partition $\boldsymbol{T}_j\boldsymbol{D}_j\boldsymbol{T}_j^T$ into two parts:

$$\boldsymbol{T}_j\boldsymbol{D}_j\boldsymbol{T}_j^T = \boldsymbol{N}_j^\times + \boldsymbol{N}_j$$

where

$$\boldsymbol{N}_j^\times = J_j^\times \boldsymbol{n}_j\boldsymbol{n}_j^T + m_j\boldsymbol{r}_j\boldsymbol{r}_j^T$$

$$\boldsymbol{N}_j = \frac{1}{2}J_j(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T - 2\boldsymbol{n}_j\boldsymbol{n}_j^T)$$

where J_j is the moment of inertia of the reaction wheel about it's axis of rotation, $J_j^\times$ is the moment of inertia about an axis orthogonal to the rotational axis, and

$$\boldsymbol{n}_j = \begin{bmatrix} \underline{n}_j \\ 0 \end{bmatrix}$$

$$\boldsymbol{r}_j = \begin{bmatrix} \underline{r}_j \\ 1 \end{bmatrix}$$

With this partitioning we note that

$$\boldsymbol{R}(\underline{s}_j)\boldsymbol{N}_j^\times \equiv \mathbf{0}$$

and since,

$$\boldsymbol{R}(\underline{v}_j) = \boldsymbol{R}(\underline{v}_3) + \boldsymbol{R}(\underline{s}_j)\dot{q}_j$$

we get:

$$K_j = \frac{1}{2}TR\{\boldsymbol{R}(\underline{v}_3)\boldsymbol{N}_j^\times\boldsymbol{R}(\underline{v}_3)^T\} + \frac{1}{2}TR\{\boldsymbol{R}(\underline{v}_j)\boldsymbol{N}_j\boldsymbol{R}(\underline{v}_j)^T\}$$

From equation 1:

$$\begin{aligned} l_j &= \frac{\partial K}{\partial \dot{q}_j} \\ &= TR\{\boldsymbol{R}(\underline{s}_j)\boldsymbol{T}_j\boldsymbol{D}_j\dot{\boldsymbol{T}}_j^T\} \\ &= J_j(\boldsymbol{n}_j^T\omega_3 + \dot{q}_j) \end{aligned}$$

where ω_3 is the angular velocity of the Base.

$$\omega_3 = \sum_{i=1}^{3}\underline{n}_i\dot{q}_i$$

Direct expansion reveals that:

$$l_j^2 = J_jTR\{\boldsymbol{R}(\underline{v}_j)\boldsymbol{N}_j\boldsymbol{R}(\underline{v}_j)^T\}$$

Thus, we can write:

$$K_j = \frac{1}{2}TR\{\boldsymbol{R}(\underline{v}_3)\boldsymbol{N}_j^\times\boldsymbol{R}(\underline{v}_3)^T\} + \frac{1}{2}J_j^{-1}l_j^2$$

Finally, we note that

$$\boldsymbol{E}_j = \boldsymbol{T}_3^{-1}\boldsymbol{N}_j^\times(\boldsymbol{T}_3^{-1})^T$$

is a constant matrix. We can therefore write:

$$K_j = \frac{1}{2}TR\{\dot{\boldsymbol{T}}_3\boldsymbol{E}_j\dot{\boldsymbol{T}}_3^T\} + \frac{1}{2}J_j^{-1}l_j^2$$

which is the desired result.

This allows us to rewrite the total kinetic energy in terms of the generalized momentum of the reaction wheels. We obtain,

$$K = \sum_{i=3}^{n+3}\frac{1}{2}TR\{\dot{\boldsymbol{T}}_i\underline{\boldsymbol{D}}_i\dot{\boldsymbol{T}}_i^T\} + \sum_{j=n+4}^{n+6}\frac{1}{2}J_j^{-1}l_j^2 \tag{5}$$

where

$$\underline{\boldsymbol{D}}_i = \begin{cases} \boldsymbol{D}_i + \boldsymbol{E}_{n+4} + \boldsymbol{E}_{n+5} + \boldsymbol{E}_{n+6} & \text{if } i = 3 \\ \boldsymbol{D}_i & \text{if } i \neq 3 \end{cases}$$

It is of some interest to note that $\underline{\boldsymbol{D}}_3$ is the 4×4 counterpart of the spatial articulated moment of inertia matrix, [12].

Note, for $k \leq 3$,

$$\begin{aligned} \frac{\partial l_j}{\partial \dot{q}_k} &= TR\{\boldsymbol{R}(s_j)\boldsymbol{T}_j\boldsymbol{D}_j\boldsymbol{T}_j^T\boldsymbol{R}(s_k)^T\} \\ &= J_j(\boldsymbol{n}_j^T\boldsymbol{n}_k) \end{aligned}$$

and

$$\frac{\partial l_j}{\partial q_k} = J_j\boldsymbol{n}_j^T\dot{\boldsymbol{n}}_k = J_j\boldsymbol{n}_j(\omega_k \times \boldsymbol{n}_k)$$

So that,

$$\frac{d}{dt}\left(\frac{\partial l_j}{\partial \dot{q}_k}\right) - \frac{\partial l_j}{\partial q_k} = J_j\dot{\boldsymbol{n}}_j^T\boldsymbol{n}_k = J_j(\omega_3 \times \boldsymbol{n}_j)^T\boldsymbol{n}_k$$

For the manipulator joints, $n + 4 > k > 3$,

$$\frac{\partial l_j}{\partial \dot{q}_k} = \frac{\partial l_j}{\partial q_k} = \mathbf{0}$$

2.2 Elimination of Base Velocity

The form of the equation for the kinetic energy of the system given in equation 5 is defined in terms of the velocities relative to the inertial coordinate frame. Our objective in this section is to rewrite this equation in terms of velocities relative to Floating Reference Frame. That is, we will eliminate the term $\dot{p}_0$ found in equation 5. This is done by rewriting this equation in terms of the velocity of the system center of mass.

We begin by noting that:

$$\boldsymbol{T}_i = \boldsymbol{T}_0\boldsymbol{A}_i$$

where A_i is the homogeneous transform of link i coordinates with respect to the Floating Reference Frame. Thus,

$$\begin{aligned}\dot{T}_i &= \dot{T}_0 A_i + T_0 \dot{A}_i \\ &= \dot{p}_0 e^T + \dot{A}_i \qquad (6)\end{aligned}$$

where $e = [0\ 0\ 0\ 1]^T$. Note that:

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & p_0 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & \end{bmatrix}$$

The position of the center of mass of link j is given by:

$$p_j^c = T_j r_j^c$$

where, $r_j^c = constant$, is the location of the center of mass of link j with respect to link j coordinates. This is illustrated in figure 1. Note that r_3^c is the center of mass of the combination of the Base and the three reaction wheels. From equation 6 we get:

$$\dot{p}_j^c = \dot{T}_j r_j^c = \dot{p}_0 + \dot{A}_i r_j^c$$

By definition of the system center of mass, we have:

$$\begin{aligned} m_T \dot{p}^c &= \sum_{j=3}^{n+3} m_j \dot{p}_j^c = \sum_{j=3}^{n+3} m_j \dot{T}_j r_j^c \\ &= \sum_{j=3}^{n+3} m_j \dot{p}_0 + \sum_{j=3}^{n+3} m_j \dot{A}_j r_j^c \\ &= m_T \dot{p}_0 + \sum_{j=3}^{n+3} m_j \dot{A}_j r_j^c \end{aligned}$$

where m_j is the mass of link j and m_T is the total mass of the system. So the linear velocity of the Floating Frame is:

$$\dot{p}_0 = -\sum_{j=3}^{n+3} \frac{m_j}{m_T} \dot{A}_j r_j^c + \dot{p}^c$$

Substituting this into equation 6 gives:

$$\begin{aligned} \dot{T}_i &= \dot{p}^c e^T + \dot{A}_i - \sum_{j=3}^{n+3} \frac{m_j}{m_T} \dot{A}_j r_j^c e^T \\ &= \dot{p}^c e^T + \sum_{j=3}^{n+3} \dot{A}_j C_{ij} \end{aligned}$$

where

$$C_{ij} = \begin{cases} I - \frac{m_j}{m_T} r_j^c e^T & \text{if } i = j \\ -\frac{m_j}{m_T} r_j^c e^T & \text{if } i \neq j \end{cases}$$

Substituting this into equation 5 gives:

$$\begin{aligned} K &= \sum_{i=3}^{n+3} \frac{1}{2} TR\{\dot{T}_i \underline{D}_i \dot{T}_i^T\} + \sum_{j=n+4}^{n+6} \frac{1}{2} J_j^{-1} l_j^2 \\ &= \sum_{i=3}^{n+3}\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{A}_j C_{ij} \underline{D}_i C_{ik}^T \dot{A}_k^T\} \\ &+ \sum_{i=3}^{n+3}\sum_{j=3}^{n+3} \frac{1}{2} TR\{\dot{A}_j C_{ij} \underline{D}_i e (\dot{p}^c)^T\} \\ &+ \sum_{i=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{p}^c e^T \underline{D}_i C_{ik}^T \dot{A}_k^T\} \\ &+ \sum_{i=3}^{n+3} \frac{1}{2} TR\{\dot{p}^c e^T \underline{D}_i e (\dot{p}^c)^T\} \\ &+ \sum_{j=n+4}^{n+6} \frac{1}{2} J_j^{-1} l_j^2 \end{aligned}$$

and since,

$$\sum_{i=3}^{n+3} C_{ij} \underline{D}_i e (\dot{p}^c)^T = 0$$

We obtain,

$$K = \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{A}_j U_{jk} \dot{A}_k^T\} + \sum_{j=n+4}^{n+6} \frac{1}{2} J_j^{-1} l_j^2 + g \qquad (7)$$

where

$$\begin{aligned} U_{jk} &= \sum_{i=3}^{n+3} C_{ij} \underline{D}_i C_{ik}^T \\ &= \begin{cases} \underline{D}_j - \frac{m_j^2}{m_T} r_j^c (r_j^c)^T & \text{if } j = k \\ -\frac{m_j m_k}{m_T} r_j^c (r_k^c)^T & \text{if } j \neq k \end{cases} \\ g &= \sum_{i=3}^{n+3} \frac{1}{2} TR\{\dot{p}^c e^T \underline{D}_i e (\dot{p}^c)^T\} \\ &= \frac{1}{2} m_T (\dot{p}^c)^T \dot{p}^c \end{aligned}$$

2.3 Lagrange's Equation

Lagrange's equation is used to obtain the dynamic equations of motion.

$$\tau_i = \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_i} - \frac{\partial K}{\partial q_i}$$

where τ_i is the actuator torque if a rotational joint or actuator force if a translational joint. For the reaction wheels, this equation is particularly easy to evaluate since the position variables are cyclic. That is:

$$\frac{\partial K}{\partial q_i} = 0$$

So that,

$$\tau_i = \frac{dl_i}{dt}$$

where τ_i is the actuator torque of the $i-th$ reaction wheel.

For the remaining variables, we note that:

$$\frac{\partial \dot{A}_j}{\partial \dot{q}_i} = \frac{\partial A_j}{\partial q_i}$$

and

$$\frac{d}{dt}\left(\frac{\partial A_j}{\partial q_i}\right) = \frac{\partial \dot{A}_j}{\partial q_i}$$

Hence,

$$\begin{aligned} \frac{\partial K}{\partial \dot{q}_i} &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\frac{\partial A_j}{\partial q_i} U_{jk} \dot{A}_k^T\} \\ &+ \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{A}_j U_{jk} (\frac{\partial A_k}{\partial q_i})^T\} \\ &+ \sum_{j=n+4}^{n+3} (\frac{1}{J_j} l_j \frac{\partial l_j}{\partial \dot{q}_i}) \end{aligned}$$

$$= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \dot{A}_j}{\partial \dot{q}_i} U_{jk} \dot{A}_k^T\} + \sum_{j=n+4}^{n+3} (\frac{1}{J_j} l_j \frac{\partial l_j}{\partial \dot{q}_i})$$

since $U_{kj}^T = U_{jk}$.

Similarly:

$$\begin{aligned} \frac{\partial K}{\partial q_i} &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\frac{\partial \dot{A}_j}{\partial q_i} U_{jk} \dot{A}_k^T\} \\ &+ \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{A}_j U_{jk} (\frac{\partial \dot{A}_k}{\partial q_i})^T\} \\ &+ \sum_{j=n+4}^{n+3} (\frac{1}{J_j} l_j \frac{\partial l_j}{\partial q_i}) \\ &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \dot{A}_j}{\partial q_i} U_{jk} \dot{A}_k^T\} \\ &+ \sum_{j=n+4}^{n+3} (\frac{1}{J_j} l_j \frac{\partial l_j}{\partial q_i}) \end{aligned}$$

since $U_{kj}^T = U_{jk}$.

Therefore, the equations of motion are:

$$\tau_i = \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial A_j}{\partial q_i} U_{jk} \ddot{A}_k^T\} + u_i$$

where

$$u_i = \begin{cases} n_i^T u & \text{if } i \le 3 \\ 0 & \text{if } 3 < i \le n+3 \end{cases} \tag{8}$$

and

$$u = \sum_{j=n+4}^{n+6} \frac{d}{dt}(n_j l_j) = \sum_{j=n+4}^{n+6} (n_j \tau_j + \omega_3 \times n_j l_j) \tag{9}$$

Since there is no external torque applied to the Base, the τ_i are zero for $i < 3$. Therefore, to simplify the development of the controller we make the following definitions. For $0 < i < n+4$ let,

$$\rho_i = \tau_i - u_i \tag{10}$$

The equations of motion then become, for $0 < i < n+4$:

$$\rho_i = \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial A_j}{\partial q_i} U_{jk} \ddot{A}_k^T\} \tag{11}$$

For the remainder of this paper, we will consider ρ_i to be the inputs to the system. If the ρ_i are given, the actuator inputs τ_j can be obtained from equations 8 through 10.

3 Adaptive Controller

In this section we present the control law and adaptation law so that the system tracts onto the desired joint trajectory. Global asymptotic stability is proven and a recursive formulation of the controller is provided for computational efficiency.

3.1 Method of Control

The controller is a modified version of an inverse dynamic controller with adaptation. Let q be an $(n+3) \times 1$ vector of the joint positions which includes the three orientation angles of the Base and the n joint angles of the manipulator. We start by defining the variable, $\dot{\tilde{q}}$ in terms of the position and velocity errors, $q_e = q - q_d$. Let:

$$\dot{\tilde{q}} = \dot{q}_e + \lambda q_e \tag{12}$$

where λ is a positive definite diagonal matrix with positive diagonal components λ_i and q_d is the desired value of q. Thinking of $\dot{\tilde{q}}$ as an input, this defines an exponentially stable and strictly proper transfer function between $\dot{\tilde{q}}$ and q_e. The method of control is to select the control law and adaptation law such that $\dot{\tilde{q}}$ is an L^2 function. It can be shown that, [13]:

$$if\ \dot{\tilde{q}} \in L^2\ then \begin{cases} q_e \in L^2 \cap L^\infty \\ \dot{q}_e \in L^\infty \\ q_e \text{ is continuous} \\ q_e(t) \to 0 \text{ as } t \to \infty \end{cases}$$

Thus, we have proven that the position and velocity tracking errors have converged to zero if we can show that $\dot{\tilde{q}}$ is an L^2 function.

A judicious choice of the norm of $\dot{\tilde{q}}$ is a critical part of the method. The following norm $\tilde{K}$ is an appropriate choice:

$$\tilde{K} = \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{\tilde{A}}_j U_{jk} \dot{\tilde{A}}_k^T\} > 0 \quad \forall \dot{\tilde{q}} \ne 0$$

where

$$\dot{\tilde{A}}_k = \sum_{i=1}^{k} \frac{\partial A_k}{\partial q_i} \dot{\tilde{q}}_i$$

The time derivative of $\tilde{K}$ is:

$$\begin{aligned} \frac{d\tilde{K}}{dt} &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\ddot{\tilde{A}}_j U_{jk} \dot{\tilde{A}}_k^T\} \\ &+ \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} \frac{1}{2} TR\{\dot{\tilde{A}}_j U_{jk} \ddot{\tilde{A}}_k^T\} \\ &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\dot{\tilde{A}}_j U_{jk} \ddot{\tilde{A}}_k^T\} \end{aligned}$$

3.2 The Controller

From equation 12 we get:

$$\dot{\hat{q}} = \dot{q}_d + \lambda q_e \tag{13}$$

where $\dot{\hat{q}} = \dot{q} - \dot{\tilde{q}}$. From this we get:

$$\ddot{\hat{q}} = \ddot{q}_d + \lambda \dot{q}_e \tag{14}$$

Defining:

$$\dot{\hat{A}}_k = \sum_{i=1}^{k} \frac{\partial A_k}{\partial q_i} \dot{\hat{q}}_i$$

Then, the control law is:

$$\rho_i = \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}\hat{U}_{jk}(\ddot{\tilde{\boldsymbol{A}}}_k - \gamma\dot{\tilde{\boldsymbol{A}}}_k)^T\} \tag{15}$$

and the adaptation law is:

$$\dot{\hat{\boldsymbol{U}}}_{jk} = -\alpha_{jk}\dot{\tilde{\boldsymbol{A}}}_j^T(\ddot{\tilde{\boldsymbol{A}}}_k - \gamma\dot{\tilde{\boldsymbol{A}}}_k) \tag{16}$$

where γ and α_{jk} are positive constants, and $\hat{\boldsymbol{U}}_{jk}$ is current estimate of $\boldsymbol{U}_{jk}$. Note that only $\ddot{\tilde{\boldsymbol{q}}}$ and $\dot{\tilde{\boldsymbol{q}}}$ are required for control and not $\hat{\boldsymbol{q}}$. Also, the acceleration, $\ddot{\boldsymbol{q}}$, is not required.

3.3 Stability Proof

We define the nonnegative function $V(t)$:

$$V(t) = \tilde{K} + 1/2\sum_{j=3}^{n+3}\sum_{k=3}^{n+3}\alpha_{jk}^{-1}TR\{\tilde{U}_{jk}\tilde{U}_{jk}^T\} \tag{17}$$

where the α_{jk} are positive constants, and $\tilde{\boldsymbol{U}}_{jk} = \boldsymbol{U}_{jk} - \hat{\boldsymbol{U}}_{jk}$ is the error in the estimate of $\boldsymbol{U}_{jk}$.

To prove stability we first show that $\dot{V}(t) \leq 0$. We start by substituting equation 15 into the equations of motion, equation 11. This gives:

$$\begin{aligned}
\boldsymbol{0} &= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}(\boldsymbol{U}_{jk}\ddot{\boldsymbol{A}}_k^T - \hat{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T)\} \\
&+ \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}\hat{U}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\} \\
&= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}(\boldsymbol{U}_{jk}(\ddot{\boldsymbol{A}}_k^T - \ddot{\tilde{\boldsymbol{A}}}_k^T) + (\boldsymbol{U}_{jk} - \hat{\boldsymbol{U}}_{jk})\ddot{\tilde{\boldsymbol{A}}}_k^T)\} \\
&+ \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}\hat{U}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\} \\
&= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}(\boldsymbol{U}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T + \tilde{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T)\} \\
&+ \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}\hat{U}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\}
\end{aligned}$$

Multiply by $\dot{\tilde{q}}_i$ and summing over all i gives:

$$\begin{aligned}
\boldsymbol{0} &= \sum_{i=1}^{n+3}\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}(\boldsymbol{U}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T + \tilde{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T)\dot{\tilde{q}}_i\} \\
&+ \gamma\sum_{i=1}^{n+3}\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\frac{\partial \boldsymbol{A}_j}{\partial q_i}\hat{U}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\dot{\tilde{q}}_i\} \\
&= \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{(\dot{\tilde{\boldsymbol{A}}}_j\boldsymbol{U}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T + \dot{\tilde{\boldsymbol{A}}}_j\tilde{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T)\} \\
&+ \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\dot{\tilde{\boldsymbol{A}}}_j\hat{\boldsymbol{U}}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\} \\
&= \frac{d\tilde{K}}{dt} + \sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\dot{\tilde{\boldsymbol{A}}}_j\tilde{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T\} \\
&+ \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\dot{\tilde{\boldsymbol{A}}}_j\hat{\boldsymbol{U}}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\}
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{d\tilde{K}}{dt} &= -\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\tilde{\boldsymbol{U}}_{jk}\ddot{\tilde{\boldsymbol{A}}}_k^T\dot{\tilde{\boldsymbol{A}}}_j\} \\
&- \gamma\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\hat{\boldsymbol{U}}_{jk}\dot{\tilde{\boldsymbol{A}}}_k^T\dot{\tilde{\boldsymbol{A}}}_j\}
\end{aligned}$$

Where we have used the trace identity $TR\{\boldsymbol{ABC}\} = TR\{\boldsymbol{BCA}\}$ for any square matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$. Adding and subtracting $2\gamma\tilde{K}$ gives:

$$\frac{d\tilde{K}}{dt} = -\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\tilde{\boldsymbol{U}}_{jk}(\ddot{\tilde{\boldsymbol{A}}}_k - \gamma\dot{\tilde{\boldsymbol{A}}}_k)^T\dot{\tilde{\boldsymbol{A}}}_j\} - 2\gamma\tilde{K} \tag{18}$$

Thus, if there are no errors in the parameter estimates, $\tilde{\boldsymbol{U}}_{jk} = \boldsymbol{0}$, then $\tilde{K}$ satisfies the linear equation, $d\tilde{K}/dt + 2\gamma\tilde{K} = 0$ and, hence, $\tilde{K}(t) = e^{-2\gamma t}\tilde{K}(0)$ and the stability result immediately follows. However, for the case in point, the parameter values are not initially known and we must proceed further.

From equation 17 we get:

$$\dot{V}(t) = \frac{d\tilde{K}}{dt} - \sum_{j=3}^{n+3}\sum_{k=3}^{n+3}\alpha_{jk}^{-1}TR\{\tilde{U}_{jk}\dot{\hat{U}}_{jk}^T\}$$

Substituting in equation 18 gives:

$$\begin{aligned}
\dot{V}(t) &= -\sum_{j=3}^{n+3}\sum_{k=3}^{n+3} TR\{\tilde{\boldsymbol{U}}_{jk}(\ddot{\tilde{\boldsymbol{A}}}_k - \gamma\dot{\tilde{\boldsymbol{A}}}_k)^T\dot{\tilde{\boldsymbol{A}}}_j\} - 2\gamma\tilde{K} \\
&- \sum_{j=3}^{n+3}\sum_{k=3}^{n+3}\alpha_{jk}^{-1}TR\{\tilde{\boldsymbol{U}}_{jk}\dot{\hat{\boldsymbol{U}}}_{jk}^T\}
\end{aligned}$$

Substituting in the adaptation equation 16 gives:

$$\dot{V}(t) = -2\gamma\tilde{K} \leq 0$$

Hence, $0 \leq V(t) \leq V(0) < \infty$, or $0 \leq V(0) - V(t) < \infty$.

From the equivalence of finite dimensional vector space norms, there exist a positive constant β such that:

$$\dot{\tilde{\boldsymbol{q}}}^T\dot{\tilde{\boldsymbol{q}}} \leq \beta\tilde{K}$$

Therefore,

$$\begin{aligned}
\int_0^\infty \dot{\tilde{\boldsymbol{q}}}^T\dot{\tilde{\boldsymbol{q}}}dt &\leq \int_0^\infty \beta\tilde{K}dt \\
&= \frac{\beta}{2\gamma}\int_0^\infty -\dot{V}(t)dt \\
&= \frac{\beta}{2\gamma}[V(0) - V(\infty)] \\
&< \infty
\end{aligned}$$

Therefore, $\dot{\tilde{\boldsymbol{q}}}$ is an L^2 function and, hence, $\boldsymbol{q}_e$ converges to zero, which is the desired result.

3.4 Recursive Formulation of Controller

The computational efficiency of the control law is greatly improved by writing the equations in a recursive form. First we define:

$$\boldsymbol{R}(\boldsymbol{s}_k) = \boldsymbol{T}_0^{-1}\boldsymbol{R}(\underline{\boldsymbol{s}}_k)\boldsymbol{T}_0$$

This matrix $\boldsymbol{R}(\boldsymbol{s}_k)$ is simply $\boldsymbol{R}(\underline{\boldsymbol{s}}_k)$ referred to the Floating Reference Frame. The vector $\boldsymbol{s}_k$ has the same interpretation as $\underline{\boldsymbol{s}}_k$ defined in equation 4 except that all the vectors are defined with respect to the floating reference frame. Let:

$$\boldsymbol{R}(\boldsymbol{v}_j) = \boldsymbol{T}_0^{-1}(\boldsymbol{R}(\underline{\boldsymbol{v}}_k) - \boldsymbol{R}(\underline{\boldsymbol{v}}_0))\boldsymbol{T}_0 = \sum_{i=1}^{j} \boldsymbol{R}(s_i)\dot{q}_i$$

Then for $\dot{\boldsymbol{A}}_k$ we have:

$$\begin{aligned} \dot{\boldsymbol{A}}_k &= \boldsymbol{R}(\boldsymbol{v}_k)\boldsymbol{A}_k \\ \boldsymbol{R}(\boldsymbol{v}_k) &= \boldsymbol{R}(\boldsymbol{v}_{k-1}) + \boldsymbol{R}(s_k)\dot{q}_k \\ \boldsymbol{R}(\boldsymbol{v}_0) &= \mathbf{0} \end{aligned}$$

For $\dot{\tilde{\boldsymbol{A}}}_k$:

$$\begin{aligned} \dot{\tilde{\boldsymbol{A}}}_k &= \boldsymbol{R}(\tilde{\boldsymbol{v}}_k)\boldsymbol{A}_k \\ \boldsymbol{R}(\tilde{\boldsymbol{v}}_k) &= \boldsymbol{R}(\tilde{\boldsymbol{v}}_{k-1}) + \boldsymbol{R}(s_k)\dot{\tilde{q}}_k \\ \boldsymbol{R}(\tilde{\boldsymbol{v}}_0) &= \mathbf{0} \end{aligned}$$

and for $\dot{\hat{\boldsymbol{A}}}_k$:

$$\begin{aligned} \dot{\hat{\boldsymbol{A}}}_k &= \boldsymbol{R}(\hat{\boldsymbol{v}}_k)\boldsymbol{A}_k \\ \boldsymbol{R}(\hat{\boldsymbol{v}}_k) &= \boldsymbol{R}(\hat{\boldsymbol{v}}_{k-1}) + \boldsymbol{R}(s_k)\dot{\hat{q}}_k \\ \boldsymbol{R}(\hat{\boldsymbol{v}}_0) &= \mathbf{0} \end{aligned}$$

For, $\ddot{\hat{\boldsymbol{A}}}_k$:

$$\begin{aligned} \ddot{\hat{\boldsymbol{A}}}_k &= (\dot{\boldsymbol{R}}(\hat{\boldsymbol{v}}_k) + \boldsymbol{R}(\hat{\boldsymbol{v}}_k)\boldsymbol{R}(\boldsymbol{v}_k))\boldsymbol{A}_k \\ \dot{\boldsymbol{R}}(\hat{\boldsymbol{v}}_k) &= \dot{\boldsymbol{R}}(\hat{\boldsymbol{v}}_{k-1}) + \boldsymbol{R}(s_k)\ddot{\hat{q}}_k + \dot{\boldsymbol{R}}(s_k)\dot{\hat{q}}_k \\ &= \dot{\boldsymbol{R}}(\hat{\boldsymbol{v}}_{k-1}) + \boldsymbol{R}(s_k)\ddot{\hat{q}}_k \\ &+ (\boldsymbol{R}(\boldsymbol{v}_k)\boldsymbol{R}(s_k) - \boldsymbol{R}(s_k)\boldsymbol{R}(\boldsymbol{v}_k))\dot{\hat{q}}_k \\ \dot{\boldsymbol{R}}(\hat{\boldsymbol{v}}_0) &= \mathbf{0} \end{aligned}$$

These, equations are computed recursively from the Floating Reference Frame to the end-effector, link $n+3$.

Next we define:

$$\boldsymbol{F}_{jk} = \boldsymbol{A}_j\hat{\boldsymbol{U}}_{jk}(\ddot{\hat{\boldsymbol{A}}}_k - \gamma\dot{\hat{\boldsymbol{A}}}_k)^T$$

$$\boldsymbol{F}_j = \sum_{k=3}^{n+3} \boldsymbol{F}_{jk}$$

and

$$\begin{aligned} \boldsymbol{f}_i &= \sum_{j=i}^{n+3} \boldsymbol{F}_j = \boldsymbol{F}_i + \boldsymbol{f}_{i+1} \\ \boldsymbol{f}_{n+4} &= \mathbf{0} \end{aligned}$$

This equation is computed recursively from the end-effector, link $i = n+3$ to the Floating Reference Frame.

Finally, the input is computed:

$$\rho_i = TR\{\boldsymbol{R}(s_i)\boldsymbol{f}_i\} \tag{19}$$

Note that:

$$\frac{\partial \boldsymbol{A}_j}{\partial q_i} = \begin{cases} \boldsymbol{R}(s_i)\boldsymbol{A}_j & \text{if } i \leq j \\ \mathbf{0} & \text{if } i > j \end{cases}$$

4 Conclusion

An efficient algorithm for the adaptive control of a space based robot has been presented. The method makes no assumptions on the initial estimates of the inertial parameters or the initial momentum of the system. Only the position and velocities of the manipulator joints and the Base orientation angles are required by the controller.

The first part of the paper develops the dynamic equations of motion for the system. Key to the method is the use of reaction wheels to control the orientation of the Base and the elimination of the Base linear motion from the equations of motion. It was shown that the effect of the reaction wheels on the dynamics of the system can be divided into two components. The first was the component related to the generalized momentum of the reaction wheels. The remaining component can be effectively included in the dynamics by modifying the inertial properties of the Base. The linear motion of the Base was removed from the equations of motion by using the law of conservation of linear momentum. With these two transformations, the resulting form of the kinetic energy was easily utilized in Lagrange's equation to obtain the dynamic equations of motion.

The algorithm used in the implementation of the adaptive controller is a modification of that presented for terrestrial based manipulators [14]. The primary differences are due to the use of homogeneous transforms in the equations of motion. For example, inner products of vectors become inner products of matrices. Another difference is the choice of the vector norm used in the stability proof. For terrestrial based manipulators this is directly related to the total kinetic energy of the manipulator. For the space based manipulator, this was related to the total kinetic energy minus the component due to the generalized momentum of the reaction wheels and the translational kinetic energy component.

A recursive form of the control algorithm was presented for computational efficiency. The computational load is still fairly high and increases quadratically in the number of links in the system. Due to the coupling of the dynamics, there does not seem to be any way to avoid the quadratic complexity problem. Since it is known that $\boldsymbol{U}_{jk} = \boldsymbol{U}_{kj}^T$, some computational improvements could be made with a slight modification of the adaptation law such that only $\boldsymbol{U}_{jk}$ is estimated instead the current method of estimating both $\boldsymbol{U}_{jk}$ and $\boldsymbol{U}_{kj}$. However, this does not appear to produce very significant improvements. A more promising approach might be to avoid all of the coordinate transformations by referring the velocities, accelerations and forces to their own link coordinates. For example, one would compute $\boldsymbol{A}_k^{-1}\dot{\boldsymbol{A}}_k$ instead of $\dot{\boldsymbol{A}}_k$ and $\boldsymbol{A}_j^{-1}\boldsymbol{F}_{jk}(\boldsymbol{A}_k^{-1})^T$ instead of $\boldsymbol{F}_{jk}$.

A significant extension of these results would be the solution for manipulators containing closed kinematic loops, since these are the most common types of manipulators encountered in practice. This would also lead to methods for dual arm coordinated motion control and compliant motion control. Another extension would be an adaptive Cartesian coordinate controller. This allow two manipulators mounted on different bases to work in a coordinated manner.

Acknowledgment

This work was supported by the NASA sponsored center for the commercialization of space, CAMRSS, located at The Environmental Research Institute of Michigan.

References

[1] R. W. Longman, R. E. Lindberg, and M. F. Zedd, "Satellite-mounted robot manipulators - new kinematics and reaction moment compensation," *International Journal of Robotics Research*, vol. 6, no. 3, pp. 87–103, 1987.

[2] Z. Vafa and S. Dubowsky, "On the dynamics of manipulators in space using the virtual manipulator approach," in *Proceedings of IEEE International Conference on Robotics and Automation*, Raleigh, NC, 1987.

[3] Y. Nakamura and R. Mukherjee, "Nonholonomic path planning of space robots," in *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, pp. 1050–1055, Scottsdale, Arizona, 1989.

[4] Y. Nakamura and R. Mukherjee, "Nonholonomic path planning of space robots via bi-directional approach," in *Proceedings of 1990 IEEE International Conference on Robotics and Automation*, pp. 1764–1769, Cincinnati, Ohio, 1990.

[5] W. W. Hooker and G. Margulies, "The dynamical attitude equations for an n-body satellite," *The Journal of the Astronoutical Sciences*, vol. XII, pp. 123–128, Winter 1965.

[6] W. W. Hooker, "A set of r dynamical attitude equations for an arbitrary n-body satellite having r rotational degrees of freedom," *AIAA Journal*, vol. 8, no. 7, pp. 1205–1207, July 1970.

[7] J. R. Wertz, editor, *Spacecraft Attitude Determination and Control*, D. Reidel Publishing Company, Boston Massachusetts, 1978.

[8] E. Papadopoulos and S. Dubowsky, "On the nature of control algorithms for space manipulators," in *Proceedings of 1990 IEEE International Conference on Robotics and Automation*, pp. 1102–1108, Cincinnati, Ohio, 1990.

[9] Y. Umetani and K. Yoshida, "Experimental study on two-dimensional free-flying robot satellite model," in *Proceedings of the NASA Conference on Space Telerobotics*, volume 5, pp. 215–224, 1989.

[10] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Cambridge, Mass, 1981.

[11] K. Fu, R. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw Hill, New York, 1987.

[12] R. Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Norwell, Mass, 1987.

[13] C. A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, New York, 1975.

[14] M. W. Walker, "Adaptive control of manipulators containing closed kinematic loops," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 11–19, Feb. 1990.

N91-20650

The AFIT Gross Motion Control Project

M. B. Leahy Jr
Air Force Institute of Technology
Department of Electrical and Computer Engineering
WPAFB OH 45433

ABSTRACT

The objective of the Gross Motion Control project at the Air Force Institute of Technology (AFIT) Robotic Systems Laboratory is to investigate alternative control approaches that will provide payload invariant high speed trajectory tracking for non-repetitive motions in free space. Our research has concentrated on modifications to the model-based control structure. We are actively pursuing development and evaluation of both adaptive primary (inner loop) and robust secondary (output loop) controllers. In-house developments are compared and contrasted to the techniques proposed by other researchers. The case study for our evaluations is the first three links of a PUMA-560. Incorporating the principals of multiple model adaptive estimation, artificial neural networks, and Lyapunov theory into the model-based paradigm has shown the potential for enhanced tracking. Secondary controllers based on Quantitative Feedback Theory, or augmented with auxiliary inputs, significantly improve the robustness to payload variations and unmodeled drive system dynamics. This paper presents an overview of the different concepts under investigation and provides a sample of our latest experimental results.

1 Introduction

An initiative at the Air Force Institute of Technology (AFIT) Robotic Systems Laboratory is the development, analysis, and experimental evaluation of intelligent robotic manipulator control algorithms. The motivation for our research is the high degree of tracking accuracy and environmental compliance required by future aerospace applications like robotic telepresence and automated flightline maintenance. The requirement for accurate high speed tracking with variable payloads can not be satisfied by classical individual joint feedback control schemes. Advanced control concepts that utilize knowledge of manipulator system dynamics are required. Those approaches must be robust and/or adapt to variations in manipulator dynamics caused by model inaccuracies, payload variation, and environmental interaction.

The objective of the Gross Motion Control project is to investigate alternative control approaches that will provide payload invariant high speed trajectory tracking for non-repetitive motions in free space. Our research has concentrated on modifications to the model-based control structure. Techniques for improving model-based controller performance can be divided into two groups based on whether they concentrate on the feedforward or feedback portion of the algorithm. AFIT is actively pursuing development and evaluation of techniques in both areas. The test case for our studies is the first three joints of a PUMA-560. The PUMA's well known design limitations provide a challenging control system design problem. Any algorithm that performed well on PUMA will work even better on the modern designs that will inhabit future flightlines. This paper provides an overview of the concepts being investigated and presents some of our latest results. Detailed information is contained in the numerous references.

This overview is organized as follows. In section two we describe the experimental evaluation environment and the control algorithm used to provide a tracking performance baseline. Section three discusses the development and evaluation of three forms of adaptive feedforward compensation while section four serves the same function for robust feedback and auxiliary input concepts. Conclusions and on-going research are the subject of section four.

2 Experimental Environment

The need to operate on equipment designed for human maintenance focuses our efforts on controllers for vertically articulated robotic systems with high torque amplification drive systems. While the modeling of link dynamics is well understood, complete modeling of drive system dynamics is difficult, if not impossible, for geared or harmonic transmissions. The motor and transmission dynamics of high torque drive systems play a major role in manipulator system dynamics [15, 18]. Therefore, the true performance potential of advanced robotic control concepts can only be determined through experimental evaluation and analysis. The experimental evaluations performed in this study were conducted under the AFIT Robotic Control Algorithm Development and Evaluation (ARCADE) environment [15]. Unless otherwise noted the algorithm servo rate is 222 Hz.

The goal of our experimental control algorithm evaluations is to validate concepts, not produce the optimum PUMA specific algorithm. Evaluations are conducted over operational configurations that excite all the manipulator's dynamical interactions so that general conclusions about algorithm performance can be drawn. Motion from $(-50°, -135°, 135°)$ to $(45°, -90°, 30°)$ in 1.5 seconds excites all the dynamics [15]. Robustness to payload variation is evaluated by attaching a series of brass disks to the sixth link mounting flange. The additional payload produces a significant change in inertial and gravitational dynamics [17, 15].

The general form of the output torque vector (τ) for a model-based control algorithm can be divided into feedforward (τ_{ff}), feedback (τ_{fb}), and auxiliary input (τ_{ax}) components.

$$\tau = \tau_{ax} + \tau_{ff} + \tau_{fb} \qquad (1)$$

Each of the five techniques discussed in this paper modifies only one of those components. The actual algorithm that controls future robots will probably have modifications to all three components but first we must understand how they function independently.

All the algorithms were implemented on a digital computer. The delay inherent in a digital implementation is handled by using the error information from the previous sample time in the current cycle output torque calculations. A single (non-adaptive) model-based control (SMBC) algorithm with complete feedforward dynamic compensation and fixed PD gains provided the performance baseline for our evaluations.

$$\begin{aligned}\tau_{ff}(k) &= [\hat{D}(q_d(k), \hat{a}) + J_{eff}]\ddot{q}_d(k) + \hat{h}(\dot{q}_d(k), q_d(k), \hat{a}) + \quad (2)\\ &\quad B_{eff}\dot{q}_d(k) + \tau_s + \hat{g}(q_d(k), \hat{a})\end{aligned}$$

$$\tau_{fb}(k) = K_v\dot{e}(k-1) + K_p e(k-1) \quad (3)$$

$$\dot{e}(k-1) = \dot{q}_d(k-1) - e(k-1)/T_s \quad (4)$$

$$e(k-1) = q_d(k-1) - q(k-1) \quad (5)$$

where: $\hat{}$ represents modeled values, and the the feedforward and feedback components are identical to the configuration employed in previous research [15].

3 Adaptive Feedforward Compensation

Three adaptive feedforward compensation techniques are in various stages of development and evaluation. In all cases the feedback loop τ_{fb} has been fixed to the same gain set used for the SMBC baseline. Fixing the feedback allows the performance improvement from adaptation to be isolated and analyzed. All three algorithms have adaptation mechanisms that are drive by trajectory errors so they can be considered as direct forms of adaptive control. Discussion will start with the most mature algorithm, adaptive feedforward compensation based on Lyapunov theory [14].

3.1 Adaptive Model-Based Control

Slotine and Li proposed an approach to adaptive model-based control (AMBC) that uses parameter adaptation based on Lyapunov theory to compensate for model-based controller limitations [27, 28]. An excellent tutorial on adaptive model-based control based on Lyapunov theory is in [22]. Successful experimental evaluation on the MIT WAM robot [21] provided the motivation for our investigation into the feasibility of the direct adaptive model-based concept for a manipulator with: high torque amplification drive system, slower peak velocities, and variable payloads.

In our initial evaluation of the AMBC concept we implemented the first control formulation proposed in [28]. The resultant AMBC algorithm had excellent tracking performance for the zero payload case and excessive endpoint error in the presence of payload uncertainity. The adaptation mechanism was also ineffective for slow trajectories [10]. The next logical step was to implement the full sliding mode version of the Slotine and Li approach [28]. However, the inclusion of the position and velocity measurement noise into the regressor produced unacceptable levels of vibration. To eliminate that problem, and separate the performance improvement due to sliding mode feedback and parameter adaptation, we implemented a version of the "Desired Compensation Adaptive Law" [20]:

$$\hat{\theta}(k) = \int_0^{T_s} \Gamma^{-1}Y_1^T(q_d(k), \dot{q}_d(k), \ddot{q}_d(k))[(\dot{e}(k-1) + \Lambda e(k-1)] \quad (6)$$

where T_s is the sample period and the integration was accomplished using the Adams-Bashforth Two-Step method as described in [4]. The adaptation mechanism now has the capability to drive the position error asymptotically to zero and regressor dependence on actual trajectory information is eliminated. An additional implementation advantage is the ability to precompute the regressor for known trajectories [20]. The basic structure of the adaptive control law remains unchanged:

$$\tau_{ff}(k) = Y_1[q_d(k), \dot{q}_d(k), \ddot{q}_d(k)]\hat{\theta}(k) + Y_2[q_d(k), \dot{q}_d(k), \ddot{q}_d(k)]\hat{\theta}_n(k) \quad (7)$$

where Y is the regressor matrix and $\hat{\theta}_n$ contains the "known" parameters and $\hat{\theta}$ contains the estimated parameters. The regressor is based on the known structure of the manipulator system dynamics and includes reflected actuator inertias and viscous and coulomb friction [14]. All $\hat{\theta}_n$ parameters were initialized to directly correspond to the nominal values used in our previous studies [15, 18]. The Λ matrix was diagonal with components $\lambda^i = k_P^i/k_D^i$, where k_P^i and k_D^i represent the diagonal terms of the position and derivative feedback gain matrices respectively.

AMBC tuning is a very heuristic procedure which is dependent on: the manipulator, the number of adaptive parameters, and the individual components of the Γ^{-1} matrix. The simple selection of a diagonal Γ^{-1} matrix can result in improved performance or disaster. The relative magnitude of the individual Γ^{-1} elements can vary widely, and aggressively adapting certain parameters can cause instability. In order to maximize algorithm performance and maintain stability we employed a rigorous three step tuning procedure [14].

There was a definite correlation between maximum tracking performance and the size of the $\hat{\theta}$ vector. Sixteen parameters was the magic number for our implementation. An interesting observation was that the amount of parameters, degrees of freedom in the space, not their physical significance was the important factor [14]. The adaptation law uses the available degrees of freedom to find the location in the parameter space which produces the minimal overall error for the three joints. Our results are consistent with another AMBC study were the authors found they could eliminate any knowledge of viscous and coulomb friction forces from the regressor and retune the adaptation law to compensate [5]. Investigations to further explore the generality of this hypothesis are underway.

The first step in the evaluation process was to baseline our controller over the standard evaluation suite. The parameter vector $\hat{\theta}$ was initialized prior to each test to a set of nominal values based on our a priori knowledge of zero payload manipulator system dynamics [17, 15]. Figures 1-6 highlight the tracking performance for both zero and 2 Kg payloads. AMBC clearly demonstrates the ability to compensate for uncertainties in drive system dynamics and end-effector payload.

A comprehensive evaluation of AMBC capabilities is underway. Investigations into the effects of: learning, parameter initialization, and feedback gains on algorithm performance have revealed that [14]:

- A short initial zero payload training phase permits the controller to learn the unmodeled drive system dynamics and errors in nominal inertial parameters. Continual learning does not hinder the algorithm's ability to adapt to variations in operational conditions.

- Transient performance during learning can be unpredictable even after initial training.
- For maximum tracking performance the adaptation processes should not be disabled. The controller modifies the parameter set over the course of the trajectory even after the learning phase.
- While the adaptation ability of the AMBC is impressive, for maximum tracking performance there is still no substitute for good nominal parameter information.
- Instead of learning the actual values, the adaptation mechanism learns the effect of the parameters on the tracking error and reacts accordingly.
- Softer PD gains reduce the robustness of any model-based algorithm to payload uncertainity and model mismatch. AMBC was no exception. However, the ability to learn nullifies the high gain advantage.

The performance of an AMBC algorithm should be carefully monitored over the expected operational range to assure that transients are within specifications. The desired trajectories should also be checked for actuator constraints such as saturation or jerk limitations. Either of those constraints can produce tracking instability.

3.1.1 Multiple Model-Based Control

An alternative to the Lyapunov based approach is the use of stochastic estimation/adaptation techniques. In addition to providing a fast means of parameter adaptation the stochastic approach explicitly accounts for the numerous sources of noise and uncertainty in a real physical system. Multiple Model Adaptive Estimation (MMAE) is a Bayesian estimation approach that employs multiple Kalman filters to quickly and accurately estimate parameters in the presence of noise and uncertainty. By combining the principles of MMAE and model-based control a powerful new form of adaptive model-based control was developed [13].

The Multiple Model-Based Control (MMBC) technique utilizes knowledge of nominal plant dynamics and principles of Bayesian estimation to provide a high degree of tracking accuracy in uncertain payload configurations. The MMBC algorithm is formed by augmenting a model-based controller with a form of MMAE. The MMAE algorithm is tuned to provide an estimate of the payload parameter ($\hat{a}$). The model-based controller combines the a priori knowledge of nominal structure with the parameter estimate to produce the multiple models of the robot dynamics required to maintain tracking accuracy.

The basic premise of the MMAE technique is that the continuous parameter vector $\mathbf{a}$ can be discretized into a finite set of possible vector values, $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K)$. The discretization of $\mathbf{a}$ must be large enough that there is a discernible difference between the models but not so large as to induce unacceptable errors in the estimate. The MMAE is composed of K Kalman filters running in parallel, each of whose plant models is based upon an assumed parameter $\mathbf{a}_k$. At the ith sample time, the measurement is passed to each of the filters. The residuals generated by the **K** filters are used to calculate the hypothesis conditional probabilities. These probabilities are used as weighting factors to generate $\hat{\mathbf{a}}$. Additional information about the principles of Multiple Model Adaptive Estimation can be found in [13, 19].

Figure 7 provides a sample of the experimental error profiles for the MMBC technique. The servo period for those evaluations was 100 Hz. Experimental evaluations have validated the simulation studies and clearly demonstrated the algorithm's potential to adapt to payload variations [13]. The MMBC approach is the most computationally complex algorithm we have evaluated, and the level of tuning difficulty is on the same order of magnitude as the Lyapunov technique. Additional research is required to determine any advantages that this method may have over the neural network or Lyapunov based concepts.

3.2 Neural Network Payload Estimation

Our concept for integrating the principles of artificial neural networks and model-based control was initially developed and experimentally evaluated for the relatively simple motions of a single vertically articulated joint [8]. Previous experiments examined the Adaptive Model-Based Neural Network Controller (AMBNNC) with varying payloads, initial conditions, and payload update rates. Those experiments showed that a Neural Network Payload Estimation (NNPE) algorithm can quickly and accurately identify payload variations from manipulator tracking error patterns. The three DOF extension was not a trivial extrapolation of our initial research and provides valuable insights into the utilization and training of ANNs for robotic control [12].

Both the MMBC and AMBNNC algorithm development started with the assumption that a reasonably accurate model of system dynamics was available. If no a priori model information is available off-line techniques can be employed to determine one [9]. Neural Network Payload Estimation (NNPE) provides a mechanism by which the payload dependence of the model-based control paradigm is reduced [8, 12]. The Adaptive Model-Based Neural Network Controller (AMBNNC) uses the output of a NNPE to adapt the feedforward dynamic compensation torques to payload variation or other disturbances that might increase tracking error. The feedforward compensation is identical to Equation (2) with the provision that the $\hat{a}$ values are now the payload parameter vector estimate produced by the NNPE.

The particular form of NNPE currently being investigated uses multilayer perceptron (MLP) artificial neural networks (ANNs) to determine the payload mass parameter. One neural network is trained and used for each individual update time of the trajectory. The neural networks consisted of (6) input nodes, (12) nodes in each of two hidden layers, and (5) output nodes. Training was performed using the same techniques and performance measurements as for the single link case [8]. To generate a representative set of training data for the multi-joint NNPE, the manipulator was run through the 3 DOF test trajectory ten times for each payload condition producing 121 training exemplars [12]. Instead of four payload payload classes with only positive payload variation the multi-joint NNPE was trained for five payload classes representing negative two to positive two kilogram variations. The step size remained at one kilogram and the desired value was still 0.9 for the actual class. Trained networks were tested in feedforward operation using vectors of position information not previously seen by the networks. Accuracy and error were calculated the same as during training tests. The trained neural nets were then ready for on-line operation and evaluation.

The augmentation of a model-based controller with a NNPE algorithm definitely improves overall tracking performance, but payload invariance has not yet been obtained [12, 11]. Current research is concentrated on removing the restrictions that limit AMBNNC performance to levels noticeably below the AMBC. Alternative paradigms for training the multilayer perceptron network are under investigation and replacing the MLP with a more sophisticated ANN is under consideration. The amount of adaptive parameters will also be increased. AMBC analysis revealed that adaptive algorithm performance was strongly correlated to degrees of freedom in the parameter space. Although the current AMBNNC implementation modifies the entire payload vector only the mass parameter is adapted. The ability to eliminate the dependence on heuristic tuning and the potential displayed by the single parameter adaptation are the motivation for our continued research in this area.

4 Feedback Compensation

Inconjunction with the adaptive feedforward evaluations two forms of feedback compensation under investigation. Both methods were mature enough to be compared against the performance of the AMBC approach over the standard evaluation suite. Those evaluations show that all three methods offer a comparable level of tracking accuracy.

4.1 MBAIC

In a series of publications Seraji has presented the development of an improved Lyapunov-based Model Reference Adaptive Controller (LB-MRAC) [24, 25, 23]. His initial PUMA evaluations were conducted without feedforward adaptation over a very slow trajectory [25]. We replicated those results and then evaluated several version of the algorithm over the standard test suite [16]. Without feedforward compensation LB-MRAC tracking accuracy is inferior to SMBC and, if the PD gains are initialized to a reasonable value, the the effect of gain adaptation is negligible. The real power of the technique is in the robust properties of the auxiliary term. Apparently Seraji also came to that realization and proposed a robust technique that incorporates an adaptive gain auxiliary input, fixed PD feedback, and the nominal dynamic feedforward compensation of a model-based controller [26]. As an example of the potential from augmenting a model-based structure with an auxiliary input we implemented a version of his approach.

Model-Based Auxiliary Input Control (MBAIC) is formed by augmenting a model-based controller with an auxiliary input based on Lyapunov theory [16]. The feedforward compensation (τ_{ff}) and feedback (τ_{fb}) are not altered. No gain adaptation is employed. MBAIC produced tracking accuracy superior to the pure LB-MRAC concept and the SMBC baseline [16]. The exact form of the auxiliary input depends on the techniques employed to calculate the velocity error and perform the digital integration. A τ_{ax} input expressed as:

$$\begin{aligned}\tau_{ax}(k) &= \mu_1 w_p \frac{T_s}{2}[e(k-1)+e(k-2)] + \\ &\quad \mu_1 w_v \frac{T_s}{2}[\dot{q}_d(k-1)+\dot{q}_d(k-2)] + \\ &\quad \mu_1 w_v \frac{1}{2}[q(k-3)-q(k-1)] \qquad (8)\end{aligned}$$

accounts for the one time step delay in error information due to our digital implementation and produced the best response [16]. Application of the MBAIC on the PUMA did not exhibit any symptoms of integrator windup. Therefore the inclusion of a "σ modification" [26] in the auxiliary term would only degrade tracking accuracy.

Figures 1-6 highlight MBAIC tracking efficacy. Addition of an auxiliary input significantly enhances model-based controller tracking accuracy and eliminates the large end-point error previously associated with operation in uncertain payload configurations. MBAIC has the potential to support both high speed trajectory tracking and environmental compliance by shifting the stiffness required for accurate gross motion control to a switchable auxiliary input. The main limitation with the MBAIC concept is the tuning process.

The starting point for our MBAIC tuning was the auxiliary input design parameters specified by Seraji [25]. The amount of time devoted to arriving at those parameters is unknown but efforts to improve the tracking by increasing the w_p and w_v values only produced increased levels of vibration. We were able to improve performance slightly by selecting the μ_1 values individually for each joint [14]. Searching the parameter space for a good set of PID gains is a non trivial task and we suspect that MBAIC tuning requires a similar degree of heuristic effort.

4.2 MBQFT

Quantitative Feedback Theory (QFT) is a frequency domain design procedure which has been successfully applied to the problems of robust flight control [7, 6]. The superior performance of the QFT in those applications motivated our investigation of a robotic implementation [1, 3]. An introduction to QFT design, and a comprehensive set of references can be found in [6]. Application to a robotic system required the development of a pseudo-continuous time (PCT) analog QFT design procedure. The combination of nonlinear feedforward compensation and PCT-QFT feedback is referred to as a Model-Based Quantitative Feedback Theory (MBQFT) controller [1, 3]

Since the PUMA case study is a 3x3 system, a 3x3 QFT multiple-input, multiple-output design was used. The 3x3 system was decoupled into three equivalent MISO loops and the interactions between the joints were modeled as disturbances. The MBQFT design evaluated in this study was based on seven plant templates equally spaced over the fast standard trajectory. The nominal feedforward compensation allows a linear QFT design to be used. The robot dynamics were linearized based on a zero payload configuration. The analog design is converted to the digital domain by an exact Z-transform and proper scaling of the control law. The feedback controller for joint one was third order over third order in the z-plane, the joint two and three controllers were fourth order over fourth order. The actual feedback control torques were produced by backwards difference equations [1, 3]:

$$\begin{aligned}\tau_{fb}(k) &= A_3\hat{e}(k) + A_2\hat{e}(k-1) + A_1\hat{e}(k-2) + A_0\hat{e}(k-3) \\ &\quad -B_2\tau_{fb}(k-1) - B_1\tau_{fb}(k-2) - B_0\tau_{fb}(k-3) \quad (9)\end{aligned}$$

$$\begin{aligned}\tau_{fb}(k) &= A_4\hat{e}(k) + A_3\hat{e}(k-1) + A_2\hat{e}(k-2)A_1\hat{e}(k-3)\\ &+A_0\hat{e}(k-4) - B_3\tau_{fb}(k-1) - B_2\tau_{fb}(k-2)\\ &-B_1\tau_{fb}(k-3) - B_0\tau_{fb}(k-4) \qquad (10)\end{aligned}$$

Equation (9) was used for joint 1 while joint 2 and 3 feedback was of the form of Equation (10).

The analog design is based on instantaneous position error information. The unmodeled one sample period delay inherent in a digital implementation was accounted for with an error estimator.

$$\hat{e}(k) = e(k-1) + (\dot{q}_d(k-1) - \dot{q}(k-1)) * T_s \qquad (11)$$

The key benefit of this approach is that analog design procedures could be used while still considering the digital effects (primarily, sampling delays). The additional on-line computational requirements, as opposed to the standard PD feedback control law, are minimal.

While algorithm development may be mathematically rigorous, tuning is usually based on heuristics. The key difference between the MBQFT and the other methods is that QFT design and tuning techniques are both well defined [6]. QFT synthesis provides an excellent initial set of controller coefficients and employs classic control tradeoffs such as giving up gain for phase margin to further tune performance [1, 3]. Empirical studies have revealed that designs based on the nominal robot dynamics tend to overstate the gain requirement [2]. If the gain is high enough to cause vibration on the robot, some phase margin must be given up to decrease the gain. The initial MBQFT controller caused excessive arm vibration due to high gains. However, only three design iterations were required to achieved the level of performance shown in Figures X-Y.

The MBQFT technique provides high speed trajectory tracking performance that is robust to small payload variations and unmodeled drive system dynamics. Replacing the τ_{fb} block with feedback laws based on PCT-QFT design resulted in up to a factor of four improvement in tracking accuracy. The non-heuristic nature of the MBQFT design and the computational simple implementation makes this approach an attractive alternative for a wide range of industrial manipulators.

5 Conclusions

The Gross Motion Control project has produced a new level of understanding about the control techniques necessary to provide the high level of trajectory tracking performance required for future Air Force applications. Model-based control can be made robust to incomplete dynamics modeling and payload uncertainity and is therefore a suitable structure for intelligent control algorithms.

Incorporating a desired compensation adaptation law, robust feedback, or an auxiliary input produced a model-based controller with payload invariant tracking for the first two joints of the PUMA. Therefore, the selection of "best" concept for enhanced tracking will depend on factors other than tracking performance. The two main considerations are tuning and computation time. While the adaptive approaches are more computational intensive the ever increasing power of modern microprocessors makes small variations in algorithm complexity a mute point. However, the tuning issue is very real and can not be ignored. The MBQFT has a distinct advantage in this area that neural networks may offset. The MBQFT design and tuning procedures are mathematically well defined and can be related to the well known parameters of gain and phase margin. For that reason we recommend the MBQFT technique for industrial applications with small payload variations. The learning capabilities and compliance potential of adaptive model-based control may be more appropriate for human arm emulation.

While are results are very promising there is still research to be done in this area. Continued development and evaluation of the AMBC and AMBNNC techniques is in progress. A compliant form of AMBC is also under investigation. Techniques for replacing the entire feedforward compensator with a neural network are being developed. Once the digital control system for the Utah/MIT had is operational we will extend our gross motion control research to that platform. Comparison between PUMA and hand evaluations will highlight the effects of manipulator dynamics and actuator systems on advanced controller tracking performance.

Acknowledgements

This research was partially supported by the Armstrong Aerospace Medical Research Laboratory Robotic Telepresence program. The author would like to thank: Paul Whalen, Mark Johnson, and Sam Sablan on the various forms of adaptive feedforward compensation and the other members of the MBQFT design team: Capt Dave Bossert, Professor Gary Lamont and Dr. Isaac Horowitz.

References

[1] D. E. Bossert, G. B. Lamont, M. B. Leahy Jr., and I. M. Horowitz. Model-Based Control with Quantitative Feedback Theory. In *Proc. of IEEE Conf. on Robotics and Automation*, pages 2058–2063, 1990.

[2] D. E. Bossert, G. B. Lamont, M. B. Leahy Jr., and I. M. Horowitz. Model-Based Control with Quantitative Feedback Theory: Empirical Model Analysis. Technical Report ARSL-90-2, Air Force Inst. of Tech., Feb 1990. Dept. of Elect. and Comp. Eng.

[3] D. E. Bossert, G. D. Lamont, I. M. Horowitz, and M. B. Leahy, Jr. *Design of Discrete Robust Controllers using Quantitative Feedback Theory and a Pseudo-Continuous Time Approach, Technical Report ARSL-89-7.* AFIT, ECE Dept, WPAFB, OH 45433, Sep 1989.

[4] R. L. Burden and J. D. Faires. *Numerical Analysis, 3rd Edition.* Prindle, Webster & Schmidt, Boston, 1985.

[5] C. R. Carignan, J. M. Tarrant, and G. E. Mosier. An Adaptive Controller for Enhancing Operator Performance During Teleoperation. In *Proc. of IEEE Conf. on Systems, Man and Cybernetics*, pages 150–155, 1989.

[6] J. J. D'Azzo and C. H. Houpis. *Linear Control Systems Analysis and Design.* McGraw-Hill Book Company, New York, 1988.

[7] I. M. Horowitz. *Multivariable Flight Control Design with Uncertain Parameters (YF16CCV), AFWAL-TR-83-3036.* AFWAL, WPAFB, OH. 45433, 1982.

[8] M. A. Johnson and M. B. Leahy, Jr. Adaptive model-based neural network control. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 1990.

[9] P. K. Khosla and T. Kanade. Parameter identification of robot dynamics. *Proc of 24th IEEE CDC*, December 1985.

[10] M. B. Leahy Jr., D. E. Bossert, and P. V. Whalen. Robust Model-Based Control: An Experimental Case Study. In *Proc. of IEEE Conf. on Robotics and Automation*, pages 1982–1987, 1990.

[11] M. B. Leahy, Jr., M. A. Johnson, D. E. Bossert, and G. B. Lamont. Robust Model-Based Neural Network Control. In *IEEE Proc. of the Int. Conf. on Systems Engineering*, August 1990.

[12] M. B. Leahy, Jr., M. A. Johnson, and S. K. Rogers. Neural Network Payload Estimation: A New Concept for Adaptive Model-Based Control. Technical Report ARSL-90-5, Submitted to the IEEE Trans. on Neural Networks, Air Force Inst. of Tech., March 1990. Dept. of Elect. and Comp. Eng.

[13] M. B. Leahy, Jr. and S. J. Sablan. Multiple Model-Based Control of Robotic Manipulators: Theory and Experimentation. Technical Report ARSL-90-1. Submitted to the 1990 CDC, Air Force Inst. of Tech., Feb 1990. Dept. of Elect. and Comp. Eng.

[14] M. B. Leahy, Jr. and P. V. Whalen. Enhancements to Robotic Manipulator Trajectory Tracking. Technical Report ARSL-90-6, Submitted to the Journal of Robotic Systems, Air Force Inst. of Tech., June 1990. Dept. of Elect. and Comp. Eng.

[15] M.B. Leahy Jr. Experimental analysis of robot control: A performance standard for the puma-560. *Proc. of the IEEE 4th Int. Symposium on Intelligent Control*, pages 257–264, September 1989.

[16] M.B. Leahy Jr. *Model-Based Auxiliary Input Control:Development and Experimental Analysis, Technical Report ARSL-89-6.* ECE Dept. AFIT, WPAFB, OH 45433, Sep 1989.

[17] M.B. Leahy Jr. Dynamics Based Control of Vertically Articulated Manipulators with Variable Payloads. *Int. J. of Robot Res.*, 9(3), June 1990.

[18] M.B. Leahy Jr. and G.N. Saridis. Compensation of Industrial Manipulator Dynamics. *Int. J. of Robot Res.*, 8(4):73–84, August 1989.

[19] P. S. Maybeck. *Stochastic Models, Estimation and Control*, volume 2. Academic Press, Inc., London, 1979.

[20] A. Morando, R. Horowitz, and N. Sadegh. Digital Implementation of Adaptive Control Algorithms for Robot Manipulators. In *IEEE Proc. of the Int. Conf. on Robotics and Automation*, pages 1656–1662, May 1989.

[21] G. Niemeyer and Slotine J-J. E. Performance in adaptive manipulator control. *Proc of 27th IEEE CDC*, pages 1585–1591, December 1988.

[22] R. Ortega and M. W. Spong. Adaptive motion control of rigid robots: A tutorial. *Proc of 27th IEEE CDC*, pages 1575–1584, December 1988.

[23] H. Seraji. A new approach to adaptive control of manipulators. *ASME J. of Dyn, Systm, Meas, and Contr*, 109:193–202, September 1987.

[24] H. Seraji. Configuration control of redundant manipulators: Theory and implementation. *IEEE Trans. on Robotics and Automation*, 5(4):472–490, August 1989.

[25] H. Seraji. Decentralized adaptive control of manipulators; theory, simulation and experimentation. *IEEE Trans. on Robotics and Automation*, 5(2):183–201, April 1989.

[26] H. Seraji. Robust High-Performance Control for Robotic Manipulators. In *IEEE Proc. of the Int. Conf. on Robotics and Automation*, pages 1663–1669, May 1989.

[27] J-J. E. Slotine and W. Li. Adaptive manipulator control: A case study. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1392–1400, 1987.

[28] J-J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *Int. J. of Robot Res.*, 6(3):49–49, Fall 1987.

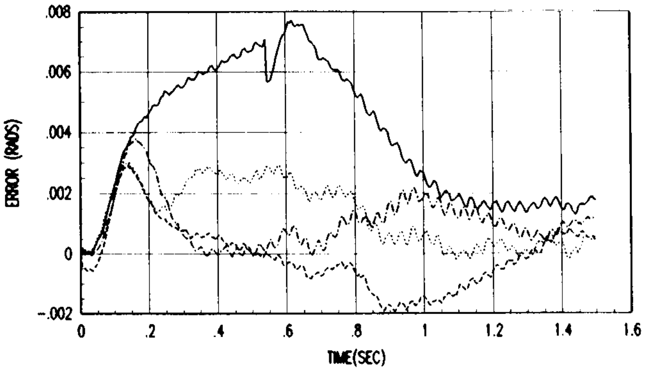


Figure 1: Joint 1 Tracking Evaluation w/o Payload

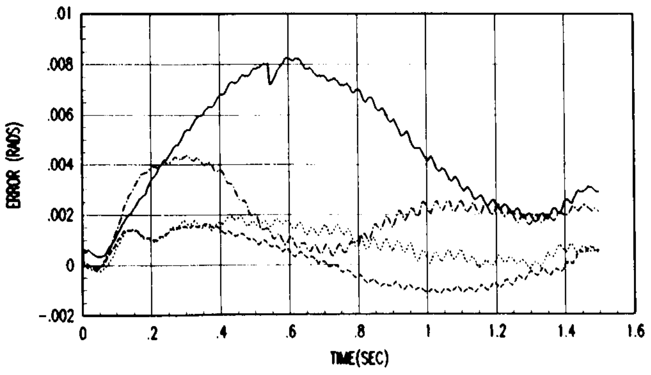


Figure 2: Joint 2 Tracking Evaluation w/o Payload

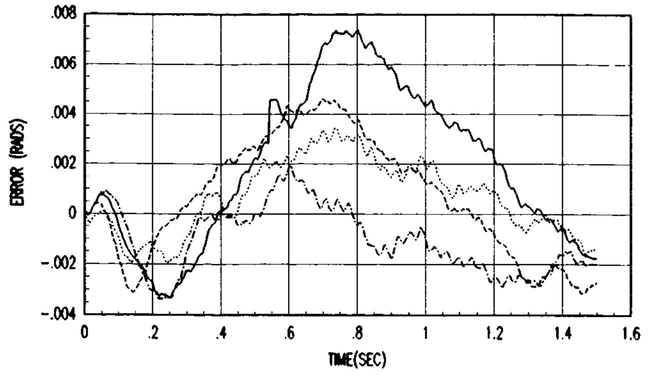


Figure 3: Joint 3 Tracking Evaluation w/o Payload

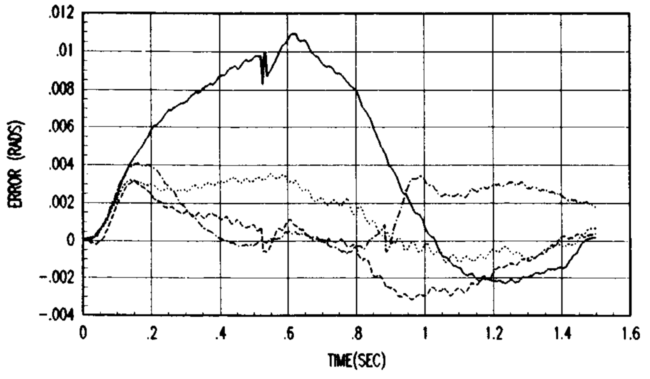


Figure 4: Joint 1 Tracking Evaluation with Payload

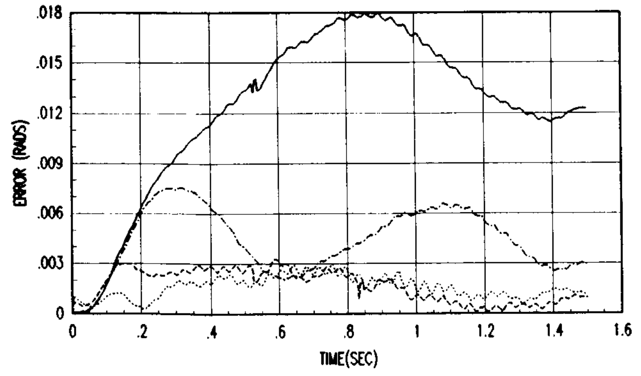


Figure 5: Joint 2 Tracking Evaluation with Payload

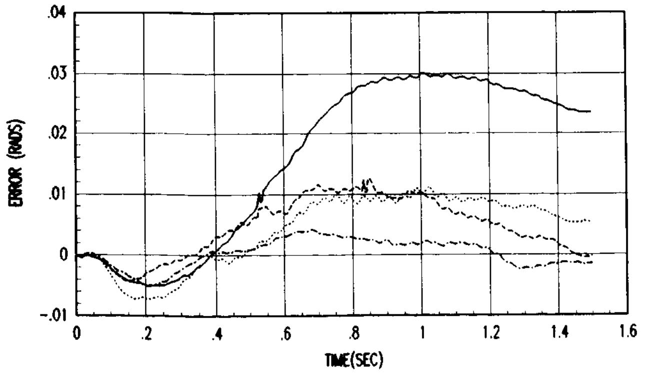


Figure 6: Joint 3 Tracking Evaluation with Payload

DATA KEY FOR FIGURES 1-6	
—	SMBC
···	MBQFT
- - -	MBAIC
- · -	AMBC

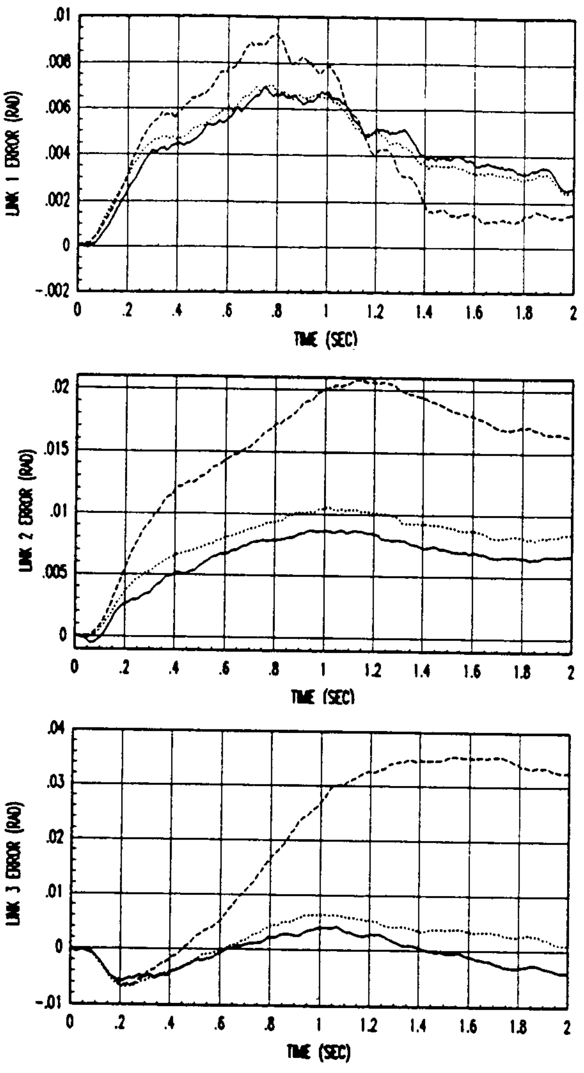


Figure 7: Tracking Evaluation with 3 Kg Payload

DATA KEY FOR FIGURE 7	
—	SMBC with Payload Info
···	MMBC
- - -	SMBC w/o Payload Info

AUTOMATION AND ROBOTICS PROGRAMMATICS

(No papers presented at this session)

MOBILITY AND EXPLORATION CONCEPTS

N91-20651

NASA PLANETARY ROVER PROGRAM

David Lavery
National Aeronautics and Space Administration
Office of Aeronautics, Exploration and Technology
Washington D.C. 20546

Roger J. Bedard Jr.
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109

ABSTRACT

The Space Exploration Initiative (SEI) announced by President Bush on July 20, 1989, the twentieth anniversary of the Apollo 11 landing, will return man to the Moon, this time to stay, and will lead to a manned mission to Mars. NASA has begun an important new technology initiative, entitled the Exploration Technology Program (ETP), which will develop the technologies needed for the SEI. Developing new technology, including a new generation of planetary rovers, is critical to the success and cost effectiveness of the President's Initiative. The Planetary Rover Project will develop the technology to enable the manned and unmanned vehicles needed for surface transportation. Surface transportation systems of many types will be required for the SEI. These include:

- o Unmanned rovers for outpost site survey and for regional exploration and science
- o Piloted rovers for transportation local to the outpost and for regional and long range manned exploration and science
- o Unmanned cargo handling, construction and mining vehicle(s)

The eventual capabilities of autonomous navigation, high mobility, low mass surface electrical power, high performance computing, high bandwidth communications with Earth, efficient thermal control, and a high level of onboard mission autonomy (eg. sample identification, acquisition and analysis, resource mining operations and outpost construction) are enabling technologies which affect the

vehicle's travel range and ultimate mission return.

The Planetary Rover Project was initiated in 1989. The emphasis of the work to date has been on autonomous navigation within the context of a high mobility wheeled vehicle at the Jet Propulsion Laboratory (JPL) and an innovative legged locomotion concept at Carnegie Mellon University (CMU). The status and accomplishments of these two efforts are discussed in the paper. First, however, this paper provides background information on the three rover types required for the SEI.

UNMANNED SCIENCE AND EXPLORATION ROVERS

The U.S. has not previously operated an unmanned roving vehicle on an extraterrestrial planetary surface. In the mid 1970's, The U.S.S.R. teleoperated (ie, using terrestrial controllers) two unmanned roving vehicles, called 'Lunakods' on the surface of the moon.

Lunakod 1 operated around 12 lunar days (almost 365 terrestrial days). The vehicle traveled a total of 10 kilometers, took many television pictures and conducted soil experiments. Lunakod 2, with twice the speed of Lunakod 1 and more experienced (and adventurous) controllers, traveled 35 kilometers in five lunar days (70 terrestrial days).

Within the SEI Program, unmanned science and exploration rovers may characterize potential outpost sites, emplace networks of science instruments, construct observatories on the far side of the Moon and perform long range exploration missions. Traverse distances of up to several kilometers per terrestrial day, through terrain containing 1 meter diameter obstacles, and a mission life of 1 to 5 years is desired for the next generation of robotic exploring vehicles.

PILOTED ROVERS

The U.S. has operated a roving vehicle on the surface of another planetary body (the Moon) with the direct involvement of a human driver. The piloted Apollo Lunar Rover Vehicle (LRV), first used in Apollo 15, provided a quantum jump in exploratory capability from the earlier Apollo missions. The LRV made it possible to travel substantial distances as well as ensuring the transportation of substantial quantities of experimental equipment and a remotely controlled television camera, which provided visual evidence of this achievement to the world. Apollo 17 (the last Apollo mission), carrying the third LRV to the moon, allowed the astronauts the longest traverse of all with a total distance of 35 km.

The next generation of piloted rovers will be satisfied by an unpressurized rover similar to the

LRV, but enhanced in range, payload and life capability. It will transport both crew and cargo about the outpost and will be used to perform human exploration and science missions up to tens of kilometers from the outpost. That rover may later be reconfigured for autonomous navigation and will perform unmanned science and exploration at distances of 1,000 kilometers from the outpost for 1 to 2 year missions.

CONSTRUCTION AND MINING VEHICLES

Technology development is required to meet the SEI needs for unmanned cargo handling, construction and mining; there is currently no technology base for extraterrestrial construction. Whereas the use of terrestrial robotics is growing in the manufacturing business, in field-oriented industries like construction, adaptation to automation technologies has been slower. Although recent progress has been made, principally, to the authors knowledge, in the United States and Japan, terrestrial cargo handling, construction and mining robotics must be considered an immature technology.

Cargo will be unloaded from a lunar excursion vehicle by a moveable gantry crane, or some other suitable device, which is teleoperated from Earth with on-site supervision by robots or a crew member. A set of interchangeable 'implements' will enable the vehicle to perform construction tasks such as excavating, relocating and smoothing regolith, and grasping and lifting objects such as boulders or structural components. The implement set will also include mining and hauling equipment for lunar soil.

NAVIGATION

Because of round trip light time, bandwidth limitation and communication channel availability delays, it is impractical to teleoperate an unmanned planetary rover from mission control on Earth. Therefore, some autonomy on the rover is needed. A highly autonomous rover capable of traveling safely over long distances for many days in unfamiliar terrain without guidance from mission control operators is beyond the present state-of-the-art. In between the extremes of teleoperation and high autonomy, various degrees of autonomy are possible. Two in particular; namely, computer aided remote driving (CARD) and semiautonomous navigation (SAN) have been identified as feasible with additional technology development

With CARD, stereo pictures from the rover are sent to mission control, where they are viewed by a human operator using a stereo display. The operator designates a safe path for the vehicle to follow as far ahead as can be seen. The plan is sent to the rover which executes the path by dead reckoning navigation aided by surface property determination sensing, maneuver level autonomous hazard detection and avoidance and expectation generation and monitoring. A new stereo pair of pictures is taken from the new position and the process repeats itself. Depending on the terrain, the rover might travel about 5 to 30 meters on each of these iterations. CARD navigation offers about a 7 km daily traverse capability on the moon and about 400 meters on Mars.

In the SAN method, local paths are planned autonomously (ie, without interaction from humans) using images obtained on the vehicle, but they are guided by global routes planned less frequently by humans in mission control. These global routes are developed from topographic map produced images obtained by an orbiting satellite or by some other means.

The sequence of operations in the portion of SAN involving mission control is as follows. As commanded from mission control, the orbiter takes a stereo pair of pictures (by taking the two pictures at different points in the orbit) of an area to be traversed. A spatial resolution equivalent to that of the rover (approximately 1 meter for a 1000 kg class unmanned rover) is desired. The pictures are sent to mission control where they are used by a human to plan an approximate route for the vehicle to follow designed to avoid large obstacles, dangerous areas and dead-ends. This route and a topographic map for the surrounding area are sent from mission control to the rover. The process repeats, as needed; perhaps once for each traverse between sites where experiments are to be done, or perhaps once per day or so on long traverses.

The sequence of operations in the portion of SAN taking place on the planetary surface is as follows. The rover views the local scene and, by using automatic stereo correlation, computes a local topographic map. This map is matched to the portion of the global map sent from mission control for purposes of position determination. The high resolution local map is analyzed by computation on the rover to determine the safe areas over which to drive. A new plan is then computed, revising the approximate route from mission control. The traverse of the revised path is simulated in order to produce sensor expectations. The expectations are used for execution monitoring and contingency planning. Using the revised path, the rover then drives, aided by surface property determination sensing, maneuver level autonomous hazard detection and avoidance and expectation generation and monitoring, a short distance (perhaps 5-10 meters), and then the process repeats. SAN navigation offers about a 24 km daily traverse capability on the moon and about 23 km on Mars.

WHEELED VEHICLE NAVIGATION DEVELOPMENT AT JPL

The major accomplishments at JPL through May, 1990 include the implementation of a wheeled rover vehicle navigation testbed, development of SAN algorithms and code, integration of SAN software onto the rover vehicle and successful feasibility demonstration.

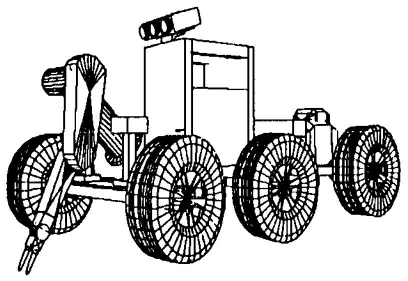


Figure 1. Robby

The construction of the wheeled rover navigation testbed, named 'Robby', was completed in December, 1989. Robby is a six-wheel, three-body articulated vehicle which offers superior mobility than conventional four-wheel, single-body vehicles. It is about 4 meters long, 1 and 1/2 meters wide and 2 and 1/2 meters high and weighs a little over 1000 kg. A commercial robot arm, for future sample acquisition experiments, is mounted on the front body. The middle body contains an electronics rack to house the onboard processors and other electronics, while serving as a mounting pedestal for the stereo camera navigation sensors. The rear body contains a commercial generator.

Over 25,000 lines of software, implementing SAN functionality, was designed, coded and integrated on Robby. In the month of May, 1990, the first continuous SAN traverse, covering a full test day, was achieved in the rough, natural terrain, arroyo test course adjacent to the JPL facility.

LEGGED VEHICLE NAVIGATION DEVELOPMENT AT CMU

The major accomplishments at CMU through May 1990 include the implementation and testing of an integrated system capable of walking with a single leg over rough terrain and the design, construction and indoor testing of the six-legged Ambler vehicle.

A prototype of an Ambler leg is suspended below a carriage that slides on rails. To walk, the system uses a laser scanner to find a clear, flat foothold, positions the leg above the foothold, contacts the terrain with the foot, and applies force enough to advance the carriage along the rails. Walking both forward and backwards, the system has traversed hundreds of meters of rugged terrain including obstacles too tall to step over, trenches too deep to step in, closely spaced rocks and sand hills.

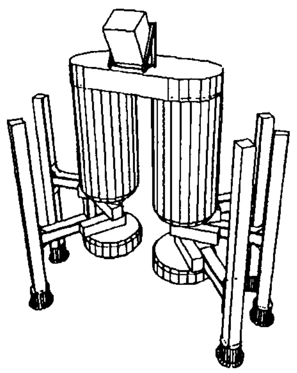


Figure 2. Ambler

The six-legged Ambler is configured to have two stacks, with six circulating legs. The actuators for body support are independent of those for propulsion. Each Ambler leg is a rotary-prismatic-prismatic orthogonal leg. The configuration enables level body motion, a circulating gait, conservatively stable gaits, high mobility and many sampling deployment options. The two shafts are connected to an arched body structure that includes four enclosures that house power generation, electronics, computing and scientific equipment. Ambler has a typical walking width of 4.5 meters, a typical walking length of 3.5 meters and a height of 4 - meters and weighs (leg and body structure) approximately 2000 kg.

SUMMARY

This paper has described the two highlights of the first year and one-half of two parts of the Planetary Rover program. Other parts of the Rover program include the development of advanced mission operations, mobility and power technology at JPL, mission operations research at Ames Research Center and piloted rover technology at the Marshall Space Flight Center. The accomplishments achieved to date represent a first step in developing the kind of machine intelligence that someday will affect how explore the universe.

ACKNOWLEDGEMENT

This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract NAS7-918 sponsored by the National Aeronautics and Space Administration and at Carnegie Mellon University under NASA grant NAGW-1175

N91-20652

An Architectural Approach to Create Self Organizing Control Systems for Practical Autonomous Robots

Helen Greiner
California Cybernetics Corporation

Introduction

For practical industrial applications, the development of trainable robots is an important and immediate objective. Therefore, we emphasize developing the type of flexible intelligence directly applicable to training. It is generally agreed upon by the AI community that the fusion of expert systems, neural networks, and conventionally programmed modules (e.g. a trajectory generator) is promising in the quest for autonomous robotic intelligence. In spite of the recent advances in all of these fields, autonomous robot development is hindered by integration and architectural problems. Some obstacles towards the construction of more general robot control systems are as follows:

1. Growth Problem- In current systems, substantial portions of the existing control software must be modified upon the addition of a new subsystem.

2. Software Generation- Currently, most software is written by people, limiting the size of code that can be created. Automatic software generation methods are premature; program writing programs are domain specific and have severe limitations.

3. Interaction with Environment- In order for the robot to properly respond to the environment, it must rely on a continuous influx of sensor data as opposed to internally stored representations. Conventional programming methods do not easily lend to massive, pipelined data processing.

4. Reliability- Most current systems are built such that single point failures cause complete system failure.

5. Resource Limitation- Current neural networks can learn most input to output functions in terms of mapping, but in case of practical problems they often take an impossibly long time to learn a function. The number of nodes or connections needed may suffer from combinatorial explosions rendering the system impossible to build.

Neural networks can be successfully applied to some of these problems. However, current implementations of neural networks are hampered by the resource limitation problem and must be trained extensively to produce computationally accurate output. Currently, there is no consensus as to the structure of an intelligent robot brain, functional break down, or interface definition. In this publication, a generalization of conventional neural nets is proposed, and an architecture is offered in an attempt to address the above problems.

Approach

The architecture that we propose consists of three components: functional groups, interfaces, and the graph describing the information flow pattern [1,2]. Each functional group performs a specific operation, and the interfaces between groups are vectors. The interconnection graph will not strongly depend on the kinematic structure of the robot. However, if a robot lacks certain sensory input, obviously the corresponding functional groups will not be present.

A functional group takes a vector as input, performs its operation, and produces an output vector. The operation of the functional group could be carried out by conventional software, hardware, or what we call a generalized network. The term generalized network describes one of the key elements in our work, and deserves detailed explanation.

A generalized network consists of two components, nodes and connections. The nodes are simply memory elements (2 byte numbers in our current implementation). The connections are able to perform mathematical operations on the node values. There is no theoretical limitation on the kind of operation that connections can perform or the number of inputs and outputs that they have (currently 16 bytes are being used). For example, a PID control servo could be a connection, where the inputs are the position setpoint and gain and the output is the commanded motor current. This method developed from a practical standpoint, to fuse advantageous properties of neural nets and table driven software. The programming is simplified because the bulk of the coding is done when the subroutine for the connection is developed. During training or operation the gains might change or connections may be created or destroyed, but this activity does not carry the risk of catastrophic software malfunction. If the task of a functional group is recognition of a situation present in sensory inputs, this group will use connections designed to best perform this task.

The architecture of the robot is defined in a hierarchical, bottom up manner, and training also occurs in this order. Each functional group is independently trained, and uses locally available information (observation of input and output vectors) to improve its behavior.

To illustrate how training occurs, we will take the example of lowest level motor control (see Figure 1). For this purpose, the sensor inputs that are directly related to motor action are separated from the rest of the sensors, and a new vector is created. A functional group is defined whose output directly drives the motors and the inputs are as follows:

- sensor vector being controlled
- a vector marking which sensor

readings should be affected

- a vector of desired sensor readings

This functional group could be realized using conventional software, if the effect of motor action is fully known to the programmer. In this case, the functional group would consist of a number of PID servos that are surrounded by conditional branches such that the servo computation is skipped if the particular sensor does not need to be affected (the enable vector). The gains in these PID servo loops would be computed based on a model of the system and modified based on observed performance. An alternative approach is to use a generalized network to carry out this control function. The tuning of the gains is automatic based on the connection's observation of the response. Assuming that the generalized network is simulated in software, the difference between it and the original software implementation is very subtle. The generalized net looks like table driven software. Later, when a custom processor is built the connection operations will be processed in parallel, making the difference more pronounced.

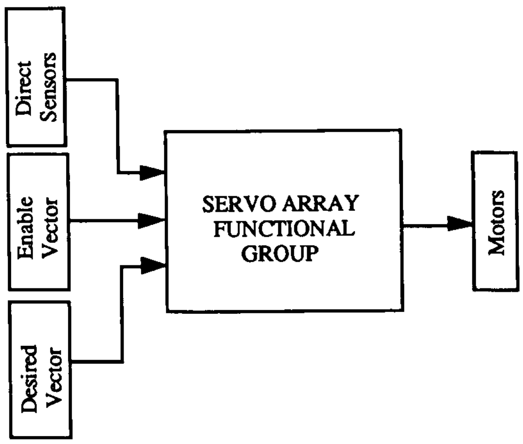


Figure 1 - Motor Servo

The advantage of using a generalized network in this instance is the relative ease of writing a list of connections. It can be seen that even this simple function of servoing low level sensor readings can be improved by various techniques that require progressively more and more computational resources. These functions can be added by adding more connections to the array.

The input to the motor servo array consists of three vectors: the direct sensor readings, the enable vector, and the desired vector. The direct sensor readings are inputs from the environment. The input nodes do not have to be physical sensor readings, nodes can be added purely to simplify later calculations. For example, in order to be able to move the tip of a robot leg along a straight trajectory in cartesian space, a new sensor node describing the x coordinate of the tip is added to the inputs. This node is calculated by conventional forward kinematic software. This is an excellent example of integrating conventional software with generalized networks. The enable vector turns individual servos on and off. This prevents servoing motors when they are not needed and can prevent two competing servos from being simultaneously active. The desired vector is a command to the motor servo group from a higher level. The objective of the motor servo group is to make the direct sensor reading as close to the desired sensor readings as possible.

The next higher level functional group is the "activity group" (see Figure 2). This group will be described in detail because it contains many elements not present in our previous example, and it has features that reappear in the higher levels. The interface between this group and the motor servo group is the desired and enable vectors which have previously been described. The input to this functional group is a vector of activities (for example, the nodes of this vector may include walking, standing, or returning to the home position), and a vector of sensory readings. The lines into and out of the activity functional group may be misleading, they in fact represent a matrix of tunable connections. The functional array contains internal nodes which all have some physical interpretation. The internal vectors are also tied together by matrices of connections. The three internal vectors used in this example are the situation vector, the vector of possible motions, and the robot motion vector. The situation vector contains nodes corresponding to certain combinations of environmental conditions. It is connected to the sensor values. A unique feature of this vector is that the nodes are competitive [3]. Strong activation of one node will inhibit activation of the others. Thus, the robot generalizes situations because a partial match of environmental conditions can cause the correct node to dominate. The next vector, the possible motions vector, contains nodes for each action such as move leg 1 up or rotate body about yaw axis. Each node is active only if the motion is possible given the current state of the robot. This prevents situations such as driving a leg while it is against a joint stop or picking up a leg when the robot's weight is on it. The last internal vector describes what motion the robot should take. Examples of nodes on this vector would be pick up leg or rotate robot body. From this vector the transformation to the desired and enable vector is straightforward.

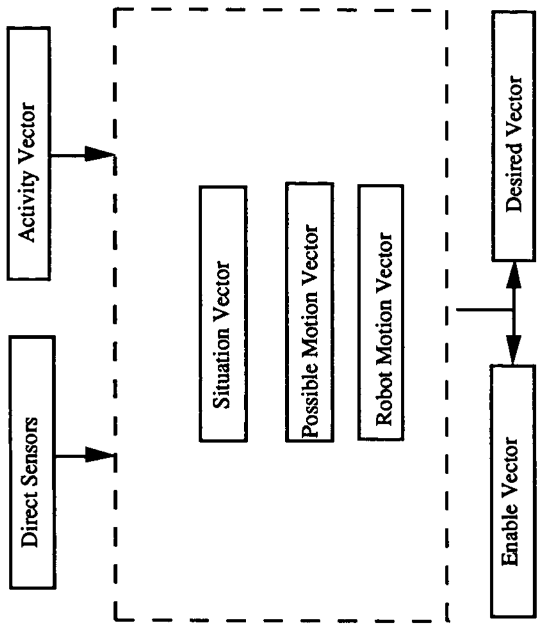


Figure 2 - Activity Functional Group

When the robot is first activated, all the connections are present. Training is a matter of the robot connections being modified to produce the correct response. Unnecessary connections are eliminated to save resources. The robot could be trained by producing random motions and seeing if any produce the correct result. However, since we know what the output vectors should be for a certain activity, another vector called a hunch is introduced. Using the hunch the robot's connections will be tuned. For example, to train the robot to walk, the node on the activity vector corresponding to walking is activated. The first hunch will activate the robot motion vector such that one leg moves forward (note: this simplified example ignores other motions that might be need to walk such as shifting body weight). The sensors at this time have caused a specific situation vector. Now unless inhibited by the possible motion vector, the connections between the active nodes on the situation and the robot motion vector will be strengthened. The next hunch may be to move one of the other legs. Again, the connections between the new situation and the motion of this leg are strengthened. This process is repeated for all the legs until if the robot is in walking mode, it has been trained what action to take given the current state of the robot. This is more valuable than simply programming the robot to move the legs sequentially because the robots actions are a function of the situation it is most nearly in.

Higher level functional groups can be added to this architecture. For example, the next level may be a "task group" in which the objective is to retrieve an object or follow a person. It is at this level that the robots begin to be useful. The bottom up approach to training of each functional group allows the higher levels to use the capabilities of the lower levels. An important point is that any improvement or additions to the lower levels improve the performance of the upper levels and don't necessitate retraining each level.

What has been described so far is one extreme of a wide spectrum of learning methods. Namely, fully hunch based learning. Learning in an intelligent system could take place totally autonomously, without the assistance of hunches. In a real learning situation, for a robot to be useful it has to simultaneously use all possible sources of information, and all beneficial learning methods. The following example will demonstrate non hunch based learning and simultaneously it will show one possible implementation of an interface between layers that facilitates smooth transition from higher level control to low level automatic execution of a task. In this learning scheme instead of behaving according to hunches the objective of learning is to maximize a scalar function called the objective function. It is assumed that the computation of this function is much simpler than carrying out the actual task. This function is either programmed into the robot by hand or somehow communicated to it. The robot control architecture generates learning as described above. To learn how to execute the task the control system has to build a list of which is the best action for every situation. The difference from the earlier case is that there is no hunch input which directly facilitates the selection of the appropriate action. The only clue as to which action is best to take is the change in the objective function. It is clearly not adequate to locally maximize the objective function with every action since several neutral or slightly adverse actions may have to be executed in a sequence before progress is made. The proposed scheme allows the robot to develop a strategy for acheiving the biggest increase in the objective function in as short a time as possible. To do this the robot builds a knowledge base that describes the consequence of its actions. This means that for every situation and every action in that situation, the robot has a prediction about what situation it will get into. Initially this data base is totally empty and the robot builds it by registering the actual sequences of situations that took place and the actions that cause them. Two distinct types of behavior are possible with this representation: goal oriented behavior and exploratory behavior. When displaying exploratory behavior the robot will try different action in situations that it has already encountered just to see the effect. On the other hand, when displaying goal oriented behavior, the robot will only chose actions which have been tried to maximize the object function as efficiently as possible. In the implementation of such a system there are two layers, reflexive and strategic. Initially, the reflexive layer is programmed with individual actions that are terminated by special situations that make the action impossible. For example, leg forward motion is terminated when the leg hits its joint limit or an obstacle. When the current motion is terminated, the reflexive layer goes idle. Detecting the idle condition the strategic layer evaluates the longterm consequence of each possible subsequent action, and choses the one deemed best in terms of the current behavior pattern (exploratory or goal oriented). Learning takes place simultaneously in both layers. The reflexive layer tries to guess what action the strategic layer will chose next. A database contains the accuracy of such guess for every situation. If the accuracy is high enough the reflexive layer will take the next action automatically (i.e. it never goes idle). In such a case the strategic layer is not involved. Learning in the strategic layer takes place by the continuous improvement of the situation action consequence database.

Conclusion

There are many advantages to creating a trainable architecture. In the introduction, obstacles towards creating a more general robot control system were listed. Now, we briefly describe how this architecture addresses these issues:

1. Growth Problem- Adding a new subsystem only effects the immediate functional group and expands it's capabilities. Addition of new sensors merely increases the number of connections in the functional array.

2. Software Generation- Software is not required to extend capabilities. Capabilities grow through training.

3. Interaction with Environment- Applicable sensory information is available at all levels of the system and the robot's action always depends on the current situation.

4. Reliability- In case of e.g. sensor failure, relevant situations are still recognized based on other sensor readings. If enhanced internal reliability is desired, the number of nodes and connections being used can be arbitrarily increased limited only by resource availability.

5. Resource Limitation- After training, the number of interconnections is reduced from O(nXn) to O(n). The connections so freed up can be reused to support learning elsewhere in the system.

We recognize that intelligent robots are a long way from being fully developed. However, practical autonomous robots can be constructed with existing technology.

References

[1] William P. Coleman et al. Modularity of Neural Network Architecture. IJCNN-90-Wash-DC, Lawrence Erlbaum Associates, Inc., 365 Broadway, Hillsdale NJ 07642, 1990.

[2] DARPA Neural Network Study. AFCEA International Press, 4400 Fair Lakes Court, Fairfax VA 22033 USA, 1990.

[3] Rumelhart, D., Hinton, G. and Williams, R. (1986). *Parallel Distributed Processing*, Vol I Cambridge, MA: MIT Press.

N91-20653

A Simple 5-DOF Walking Robot for Space Station Application

H. Benjamin Brown, Jr., Mark B. Friedman and Takeo Kanade
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
412-268-7692

ABSTRACT

Robots on the NASA space station have a potential range of applications from assisting astronauts during EVA (Extra-Vehicular Activity), to replacing astronauts in the performance of simple, dangerous and tedious tasks; and to performing routine tasks such as inspection of structures and utilities. To provide a vehicle for demonstrating the pertinent technologies, we are developing a simple robot for locomotion and basic manipulation on the proposed space station. In addition to the robot, we have developed an experimental testbed including a 1/3-scale (1.67-meter modules) truss and a gravity compensation system to simulate a zero-gravity environment.

The robot comprises two flexible links connected by a rotary joint, with 2-dof "wrist" joints and grippers at each end. The grippers screw into threaded holes in the nodes of the space-station truss, and enable it to walk by alternately shifting its base of support from one foot (gripper) to the other.

Present efforts are focused on mechanical design, application of sensors, and development of control algorithms for lightweight, flexible structures. Long-range research will emphasize development of human interfaces to permit a range of control modes from teleoperated to semiautonomous, and coordination of robot/astronaut and multiple-robot teams.

INTRODUCTION

We are developing a telerobotic Self-Mobile Space Manipulator (SM^2) for use on trusswork like that which will form the backbone of Space Station Freedom. Our design criteria have been chosen to complement the capabilities of space-suited astronauts as well as the features of robots already designated for deployment on Space Station. The SM^2 has a simple, modular, 5-DOF design for economical implementation and easy maintenance. It has low mass and is capable of safe, independent locomotion from node to node on space station trusswork, without touching the trusswork struts. The SM^2 can move autonomously, but can also be guided using various levels of telerobotic control - from high-level, goal directed commands to the lowest level of joint torque specification.

For example, under autonomous operation we envisage the robot "walking" along specified nodes on the trusswork, doing routine visual inspection, then reverting to low level teleoperation to allow an astronaut within the space station to examine and repair anomalies discovered during the inspection process. At levels of self-guidance intermediate between those just mentioned, astronaut control of semi-autonomous functions could be invoked to operate the robot to resupply and assist the Flight Telerobotic Servicer (FTS) or astronauts during EVA by bringing components or tools to a worksite. The SM^2 could be useful as an "active tether" by attaching to and positioning lights, cameras, or sub-assemblies handed to it during work at an EVA job site. During construction activities, the SM^2 should be capable of autonomous assembly operations with properly designed components.

The robot's low mass and compliant design will permit the SM^2 to move on space trusswork with minimal induction of truss vibration or disturbance to the space station's microgravity environment. These same factors, coupled with proximity sensing, also reduce the potential for the robot to accidently cause injury to space station equipment or personnel. The SM^2 system will be equally suitable for the remote construction and maintenance of other large structures to be assembled in space, including sensing platforms and reflector arrays.

We are developing a one-third scale SM^2 system on a similarly scaled model of space station trusswork. Our testbed includes a system to compensate for earthbound gravitational effects as well as three bays of trusswork supplied by StarNet Structures that are similar to NASA's design. Its nodes are full size and geometrically derived from the NASA design, but the truss struts are foreshortened to reduce the basic cubic bay side dimension from 5 meters to 1 2/3 meters (See Figure 1).

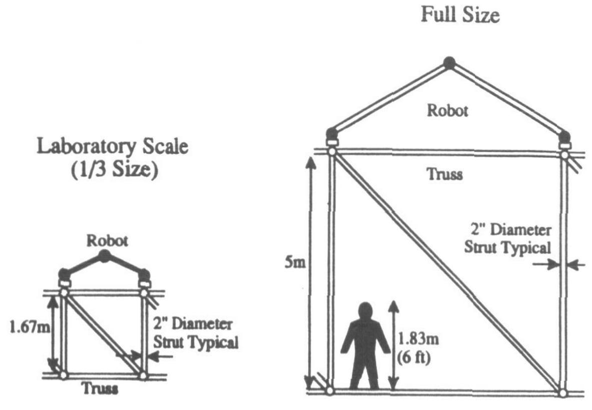

Full Size
Laboratory Scale (1/3 Size)
Robot
Truss
2" Diameter Strut Typical
1.67m
5m
1.83m (6 ft)
Robot
Truss
2" Diameter Strut Typical

Figure 1: Overall dimensions of the truss and robot are scaled to 1/3 to permit experiments in the laboratory, while local dimensions (sizes of nodes, joints and grippers) are the same to keep local behavior similar, and mechanism size workable.

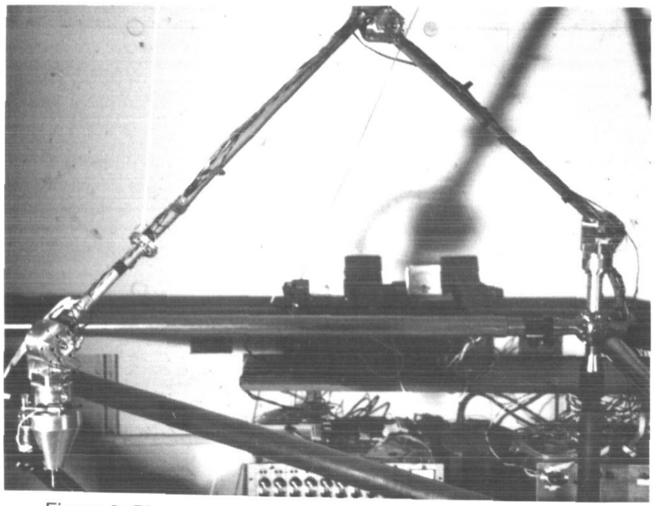


Figure 2: Photograph of the robot on 1/3-scale truss.

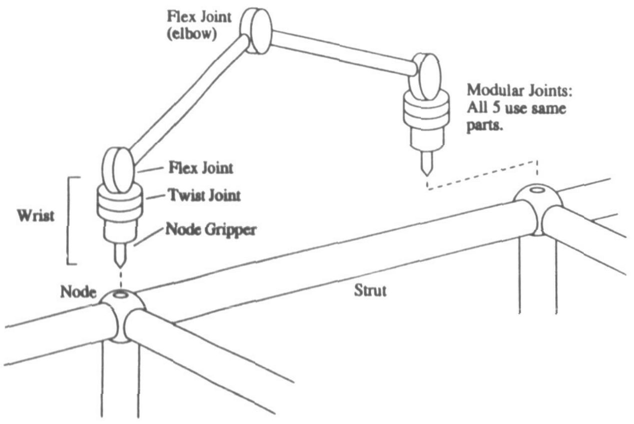


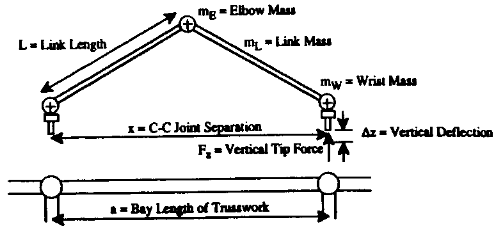
Figure 3: Schematic drawing of SM^2

ROBOT HARDWARE

The basic robot walker, shown in Figures 2 and 3, is a simple, 5-joint configuration which has the minimum size and number of degrees of freedom (dof) to permit walking on the space-station trusswork. The robot comprises a pair of slender links attached at an "elbow" flex joint, with 2-dof "wrist" joints and special grippers at both ends. The grippers screw into threaded holes in the nodes of the truss to attach the robot to the truss. The robot can span adjacent nodes which are 1.67 meters apart for our 1/3-scale laboratory robot, 5 meters for full scale. It walks by releasing one gripper, swinging to the next node and gripping; then repeating the process with the other foot. Although the robot has all its links in a plane at any time, its plane of operation can be rotated by the outboard twist joints so it can, in theory, access any unoccupied hole (26 holes per node at 45-degree spacing) of any node of the truss. With appropriate end-effectors, this configuration also permits limited manipulation capability.

The design for the laboratory robot is based on a hypothetical, full-sized, self-contained robot to be used on the Space Station; scaling rules were applied so the dynamic behavior - masses, stiffnesses, natural frequencies, linear speeds - of the scaled-down robot would be similar to that of the hypothetical one. As can be seen from Figure 1, overall dimensions of the truss and robot were reduced to 1/3, while local dimensions - of truss nodes, joints and grippers - were kept equal. This allows the testbed to be used in a laboratory of reasonable size, while mechanism are not unworkably small. Figure 4 gives some basic parameters for the scaled and full-sized designs.

The robot is designed for mobility in a zero-gravity environment, with simplicity and low mass as primary design goals. The robot is assembled from five, compact, self-contained, modular joints. As shown in Figure 5, each joint contains a DC motor, harmonic drive (60:1 or 100:1 reduction), and a potentiometer and incremental optical encoder for measuring joint angle. Joint torques are sufficient to move the robot's limbs at reasonable rates, but too low to support the robot's weight; thus it can operated only when gravitational effects are removed. Each joint weighs about 2.7 kg (1.2 lb.), and has a peak torque of 14 N-m (125 lb-in) (for 100:1 gearing) and peak speed of 5.8 radians/sec (100:1 gearing) The two links that connect the three flex joints are slender, thin-walled aluminum tubes having substantial compliance; the end-effector deflects nearly 150 mm (6 inches) under full joint torque when the robot is fully extended. The links of the 1/3-scale robot were designed to reflect the compliance of links in the full-size robot, where link mass is a significant factor. Possible improvements in the design include reduction of joint friction arising from motor brushes, bearings and gearing; and of joint backlash arising from bearing clearances. Both these factors aggravate control problems.



Parameter	Formula	1/3 Size	Full Size
Bay Length	a	1.67m	5.00m
Link Length	L	0.97m	2.91m
Link Tube Dimensions (OD × wall thickness)		19mm × 0.7mm	51mm × 1.0mm
Link Mass	m_L	0.11kg	1.23kg
Wrist Mass*	m_W	14kg	14kg
Tip Stiffness (@ x=2l)	$k=3EI/(2L)^2$	5.64 kgf/m	5.67 kgf/m
Tip Force @ Max Joint Torque	F_z	0.92 kgf	0.92 kgf
Tip Deflection @ Max Joint Torque	$\Delta z=T(2L)^2/3EI$	148mm	148mm
Lowest Natural Frequency (@ x=2l)	$\omega_N=\sqrt{(k/m_W)}$	2.19 rad/sec	2.20 rad/sec
Step Time (nominal for 180° step)	t_{180}	6.7 sec	20 sec

*Estimated for self-contained robot with auxiliary manipulators.

Figure 4: Scaled parameters for full-size hypothetical robot and 1/3-size laboratory robot.

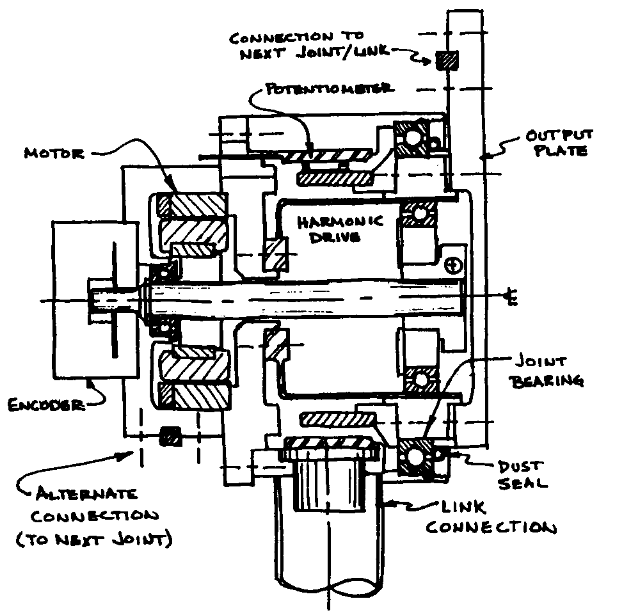


Figure 5: Joint is compact, self-contained, modular design. Joint includes a DC motor, harmonic drive reducer, position sensors and bearings.

We minimize mechanical backlash and friction, and enhance mechanical stability, with harmonic-drive gearing and four-point-contact joint bearings (Kaydon type X). Because the links are very light, we can assume the mass is concentrated at the joints, which simplifies control significantly by practically eliminating the high vibration modes associated with distributed link mass. Keeping the robot lightweight, in general, permits acceptably fast control with low torques, although joint friction is still about 10% of available peak torque.

The node gripper, the device that attaches the robot to the nodes of the trusswork, is a critical part of the design. Unlike a typical robot end-effector, it must be able to anchor the robot firmly to the nodes, because the robot's base of support shifts from one end to the other during walking; the robot depends on this attachment point to provide a precise, stable frame of reference. The node gripper includes a screw that engages the threaded holes in the nodes, a motor and gearing that drive the screw, and a potentiometer to sense the gap between the faces of the node and gripper. After the screw is fully engaged, an internal cam mechanism draws the gripper against the node with more than 1800 N (400 lb.) of force; this prevents twisting or rocking on the node, which would disturb the robot's frame of reference. In the future, we plan to develop other end-effectors for general manipulation or specific tasks such as assembly of trusswork.

GRAVITY COMPENSATION SYSTEM

The zero-gravity environment at an orbiting space station has significant impact on the design and performance of a robot. The absence of gravitational forces permits a long, spindly robot to move relatively large masses with small forces and small consumption of power. In order to perform realistic experiments on earth, we have developed a gravity compensation system that balances the more significant gravitational effects. As shown in Figure 6, the cable supporting the robot is suspended from an overhead gantry that tracks the movements of the robot in the horizontal plane using an infrared camera and robot-mounted light source. The support cable, which attaches to a spreader beam above the robot, is routed through a system of low-friction pulleys to a low-inertia counterweight. Because of the lever arrangement, the counterweight adds only 10% to the robot's "vertical inertia." Discrepancies in the compensation forces due to friction and tracking errors amount to about 1% of the robot's weight in the vertical direction and 2 - 4 % in the horizontal. With the current system, the robot can walk reliably on the top face of the trusswork. Improvements are planned to provide better horizontal tracking (reduced side forces), to reduce friction in the counterbalance system, and to permit walking on the side faces of the truss and carrying payloads.

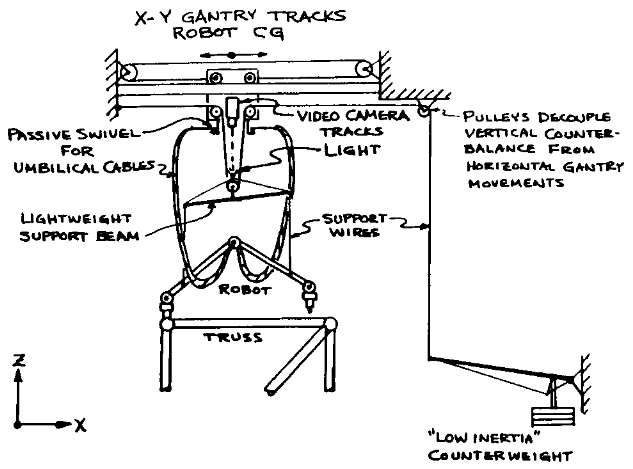

X-Y GANTRY TRACKS ROBOT CG
VIDEO CAMERA TRACKS LIGHT
PULLEYS DECOUPLE VERTICAL COUNTER-BALANCE FROM HORIZONTAL GANTRY MOVEMENTS
PASSIVE SWIVEL FOR UMBILICAL CABLES
LIGHT
LIGHTWEIGHT SUPPORT BEAM
SUPPORT WIRES
ROBOT
TRUSS
Z
X
"LOW INERTIA" COUNTERWEIGHT

Figure 6: Gravity compensation system includes a passive counterbalance system for vertical support and an active horizontal tracking system.

SENSORS

To enhance SM^2 system reliability and versatility, we strive for sensor redundancy both in the factors to be measured by sensors and in the utilization of these sensors' information. For instance, link deflection may be measured both by an internal optical system based on lateral effect diodes and by strain gages laminated to the link shell. During unconstrained limb movement, both these deflection measures - with their different ranges, resolutions, and response times - can be used for controlling limb position, while when both ends of the robot are attached to objects, these same sensors can measure the forces generated by deflection of the compliant links.

A small CCD camera head is located at each end of the robot. The video is necessary for fine human guidance during teleoperation of the robot and we try to make use of this wide bandwidth sensor for autonomous control, too. We would like to use vision for end-effector target acquisition and guidance, as well as for the direct estimation of the relative position of the two ends of the robot. The challenge for our robust use of machine vision for automatic end-effector guidance is the wide dynamic range of light intensities to be found in space. It may be impossible to simultaneously see aspects of objects illuminated by direct sunlight and in deep shadow. The contrast across shadow boundaries exceeds the dynamic range of most small imagers, so machine vision aids and algorithms must be chosen carefully. Optimally, new imaging sensor structures, with wider dynamic ranges can be fabricated using conventional VLSI techniques.

We make use of both potentiometric and incremental optical encoder information from each joint to measure joint angle and velocity. For most of the permissible rotation of each robot joint, potentiometer readout is directly proportional to joint angle, being linear to about .001 revolution (6.3 milliradians, equivalent to about 1 cm at the end-effector). By calibrating at the 90-degree positions, corresponding to the target (node) locations, our sensor accuracy at critical points is an order-of-magnitude better. However, sensor errors are overshadowed by structural deflections: elastic deflections in the links due to system dynamics and disturbances from the gravity-compensation system, and joint deflections due to backlash in the gearing and bearings. Such deflections are present to some degree even during calibration. The resultant positioning accuracy, about 1 cm, is marginal for the node-insertion task, hence our effort to use camera and other end effector-target related information.

Silicon accelerometers on the wrist (ankle) joints directly sense tip vibration and help smooth motion control at high response frequencies.

CONTROL

SM^2 robot control issues are more fully discussed in reference 6 by Ueno and Xu, et al. Three factors make control of our robot difficult: the long reach of the robot (greater than 5 meters at full size), the low joint torques available, and the compliance of the structure. Small angular deflections, due to sensor errors, backlash and structural deformation, are amplified into significant linear deflections at the robot's end-effector. Because torques are low, we want to keep joints and links light. Friction in the joints becomes a significant nonlinearity that must be dealt with. Structural compliance further increases the uncertainty in tip-position measurements and permits high-amplitude, low-frequency (around 1 hz), as well as mid-frequency (around 20 hz) vibrations in the unanchored robot structure.

There has been a great deal of interest during the last 5--10 years in the control of flexible arms (references 1, 2, and 4). Most of this has been theoretically oriented, focusing on rigorous identification and control of simple arms often with exaggerated flexibility}. Little work has been reported on application-oriented, multiple-joint systems. In contrast to most of the research in this field, our goal is not to study the control of flexible arms, but to obtain a working system. We desire to control a 5-joint, 3-dimensional robot that has substantial flexibility resulting from the necessarily lightweight design.

Control algorithms are borrowed from conventional, rigid-arm control with several modifications. We use P/D and PID controls with "gentle" input trajectories and low-pass filtering to minimize excitation. For locomotion, we employ a "coarse control" phase that uses acceleration feedback and low gains for a smooth, stable motion to the area of the target node. Once the end-effector is close to the target, the mode switches to "fine control," using higher gains and integral feedback to minimize the static error.

Real-time control is implemented digitally on an Ironics M68020 single board computer on a VME backplane, running the CHIMERA II real-time operating system. Aside from supplying a high-performance real-time kernel, CHIMERA II provides a layer of transparency between the diverse hardware and the control software (ref 5). Selecting CHIMERA II as the real-time operating system over commercially available operating systems was also motivated by its powerful multiprocessing features, which allows us to distribute the control code over multiple processors if necessary. A Sun 3/260 host workstation is used for code development and graphical displays.

HUMAN - MACHINE INTERFACE

Telerobotic control is currently based on a mouse-driven screen interface generated on a Sun 3/260. Using a hierarchical system of screen displays, an operator can choose between low-level joint position control and higher level end effector target designation. In the latter case, the computer derives and displays the sequences of end-effector motion for operator preview and approval, while in the former case, the operator directly specifies end-effector trajectories.

The challenge in design of a control station for the SM^2 robot is the non-anthropomorphic design and alternating base of support during locomotion. Dual hand controls and "state-of-the-art" helmet mounted virtual displays that are slaved to operator head movements are more appropriate for anthropomorphic robotic designs such as FTS, which has recognizable head and hand analogs. Similarly, since the SM^2 manipulator has no fixed base of support and alternates which end is attached to the space station trusswork, more conventional robot arm control interfaces are less than satisfactory. We are evolving our own human-interface control station both to satisfy our current control needs and to accomodate future requirements to coordinate the activities of multiple Self Mobile Space Manipulators working in harmony. Teleoperation is based on gestural control by an operator using a hand-held Polhemus 6-DOF pointer to guide the 5-DOF motion of the free end of SM^2 in Cartesian space. Computer mediation between the 6-DOF control and the 5-DOF robot protects against illegal motion commands, allows for scaling, indexing and calibrating robot motion, and selective axes isolation for task-specific motions (such as linear insertion in the absence of rotation, or rotational alignment without sensitivity to pitch or yaw movements of the controller.) We are developing a force reflecting hand controller that is articulated isomorphically with the SM^2 and that appropriately attaches, at alternate ends, to the control station.

CONCLUSIONS

The one-third scale Self Mobile Space Manipulator currently walks on horizontal surfaces of bays of trusswork that are assembled in our laboratory. It walks by alternately attaching to adjacent nodes under various levels of telerobotic control - ranging from autonomous multi-step moves to low-level teleoperation. Our next goals include walking over the edges of the trusswork to demonstrate 3-D locomotion capability. We intend to extend the capability of our gravity compensation system to allow us to do simple parts transportation and manipulation tasks with SM^2. We plan to do demonstration projects, in cooperation with NASA's Space Station contractors, to establish the capability of this design for solving the evolving inspection, maintenance and construction needs of Space Station Freedom.

ACKNOWLEDGMENTS

We thank Dr. Yangsheng Xu and Mr. John Dolan of CMU for their contributions to our robot control and support system control algorithms. We greatfully acknowledge the continued financial support of the Shimizu Corporation Space Projects Office as well as the technical support provided by Hiroshi and Miyuki Ueno and Tetsuji Yoshida of Shimizu Corporation.

REFERENCES

[1] T.E. Alberts, W.J. Book, and S.L. Dickerson. Experiments in augmenting active control of a flexible structure with passive damping. In *AIAA*, Reno, Nevada, January 1986. Georgia Institute of Technology.

[2] R.H. Cannon and E. Schmitz. Initial experiments on the end-point control of a flexible one-link robot. *International Journal of Robotics Research*, 3(3):62–75, Fall 1984.

[3] Scott S. Fisher. Telepresence master glove controller for dextrous robotic end-effectors. *Journal of the Society of Photo Optical Instrumentation Engineers*, 726, 1986.

[4] F. Pfeiffer and B. Gebler. A multistage-approach to the dynamics and control of elastic robots. In *Proc. 1988 IEEE International Conference on Robotics and Automation*, pages 2–8, Philadelphia, Pennsylvania, April 1988. Computer Society Press.

[5] David B. Stewart, Donald E. Schmitz, and Pradeep K. Khosla. Implementing real-time robotic systems using chimera ii. In *IEEE International Conference on Systems Engineering*, Pittsburgh, Pennsylvania, August 1990.

[6] Hiroshi Ueno and Yangsheng Xu et al. On control and planning of a space station robot walker. In *Proc. of 1990 IEEE International Conference on Systems Engineering*, Pittsburgh, Pennsylvania, August 1990.

Perception, Planning and Control for Walking on Rugged Terrain

Reid Simmons Eric Krotkov

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The CMU Planetary Rover project is developing a six-legged walking robot capable of autonomously navigating, exploring, and acquiring samples in rugged, unknown environments. To gain experience with the problems involved in walking on rugged terrain, we built a full-scale prototype leg and mounted it on a carriage that rolls along overhead rails. This paper describes issues addressed in developing the software system to autonomously walk the leg through rugged terrain. In particular, we describe the insights gained into perceiving and modeling rugged terrain, controlling the legged mechanism, interacting with the ground, choosing safe yet effective footfalls, and planning efficient leg moves through space.

1 Introduction

The CMU Planetary Rover project is constructing the Ambler, a walking robot designed for planetary exploration [2]. The configuration is a six-legged vehicle with orthogonal legs and an overlapping gait [1]. These features are designed to maximize power usage and to simplify planning and control. To meet its mission goals, the Ambler must be able to autonomously traverse rugged and often uncertain terrain, while maintaining a stable platform for its sensors and scientific equipment.

A single leg of the Ambler was built and suspended from a carriage attached to overhead rails. We developed a distributed software system that integrated perception, planning, and real-time control to autonomously walk the mechanism through a variety of obstacle courses [6, 10]. The rationale was that ideas would be easier to develop using just a single leg, and that many of the concepts would transfer to the full six-legged walker.

This paper reports on our initial experiences using the single leg of the Ambler. It focuses on the special problems encountered in perception, control and planning for rough terrain walking. In particular, we discuss the problems of modeling 3D terrain, detecting and controlling forceful interaction with terrain, and planning steps that lead to a balance between efficiency, risk, and progress of the mechanism. Readers interested in more details of the single-leg walking system should consult [6, 10].

2 Single-Leg Testbed

A single leg of the Ambler (based on an early design [2]) was built to experiment with mechanism control and system integration before committing to the fabrication of a six-legged vehicle. The leg (Figure 1) has a working radius of approximately 2.5 meters and a vertical range of travel of about 1.5 meters. The dimensions were chosen to enable the Ambler to meet its design objectives of crossing one meter wide ditches and stepping over one meter high obstacles. The leg is supported by a carriage mechanism that is mounted on a pair of rails. The carriage can roll along the rails, providing one degree of translational freedom, and the leg can rotate freely under the carriage. The support system is designed to be statically and dynamically stable, and to allow the leg to walk in a manner sufficiently similar to the Ambler so that

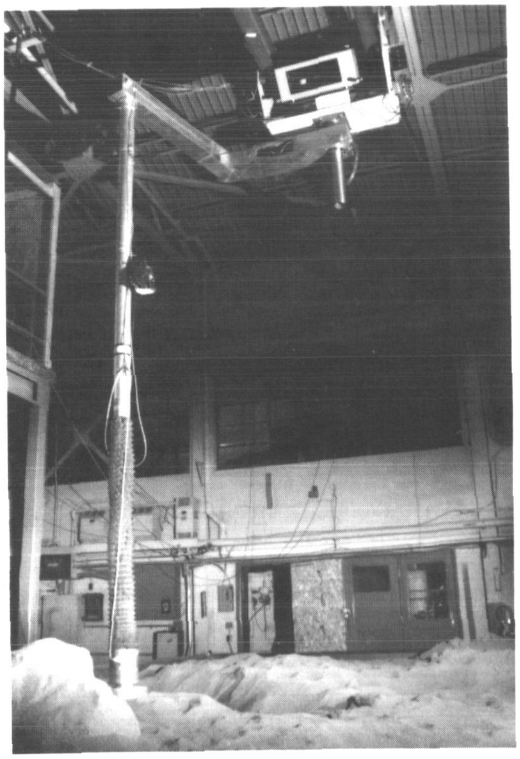


Figure 1: The Single-Leg Testbed

ideas generated could be easily transferred to the six-legged machine.

Sensors attached to the leg include a potentiometer to measure the position and velocity of the carriage along the rails, incremental and absolute encoders to measure leg positions, and two inclinometers to measure the rotation of the carriage. In addition, a six-axis force/torque sensor is attached to the bottom of the leg to measure the forces experienced by the mechanism as it moves.

A scanning laser rangefinder, manufactured by Erim, is fixed to the carriage (Figure 1). The scanner can acquire 64 by 256 pixel range and reflectance images in half a second. It digitizes to 8 bits with a range ambiguity interval of approximately 20 meters. This provides a range resolution of approximately 7.62cm. The measurements cover 80 degrees in the horizontal direction (azimuth) and 30 degrees in the vertical direction (elevation).

To provide for a variety of "Mars-like" terrains, we constructed an obstacle course below the rails measuring approximately 11 by 6 meters (Figure 2). The course is filled with over 40 tons of sand. Terrain features are introduced by resculpting the surface to form hills and trenches, and by placing objects on the sand. We have used styrofoam boulders, traffic cones, and large boxes to test the ability of the system to navigate over and around obstacles.

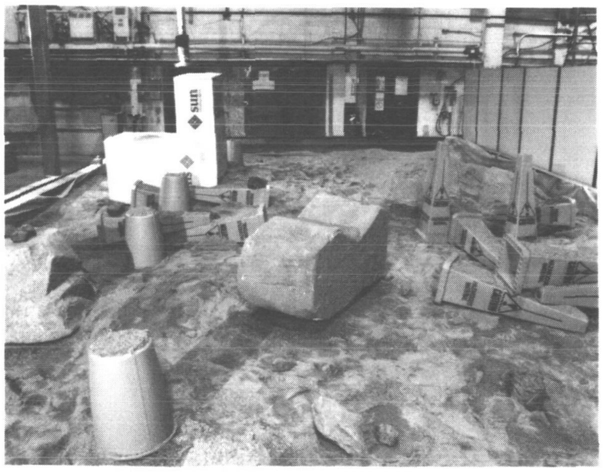


Figure 2: A Typical Arrangement of the Obstacle Course

3 Rough Terrain Walking

The single-leg walking system consists of five distributed modules (Figure 3) integrated by the centralized *Task Control Architecture (TCA)* [9, 11]. The modules communicate with one another (and with the TCA central control modules) by passing messages through the central control, which routes them to the appropriate modules and message handlers. TCA is basically a high-level robot operating system that provides utilities for building and coordinating mobile robot systems. The utilities are meant to bridge the gap between task-level planners and real-time control systems. In particular, TCA

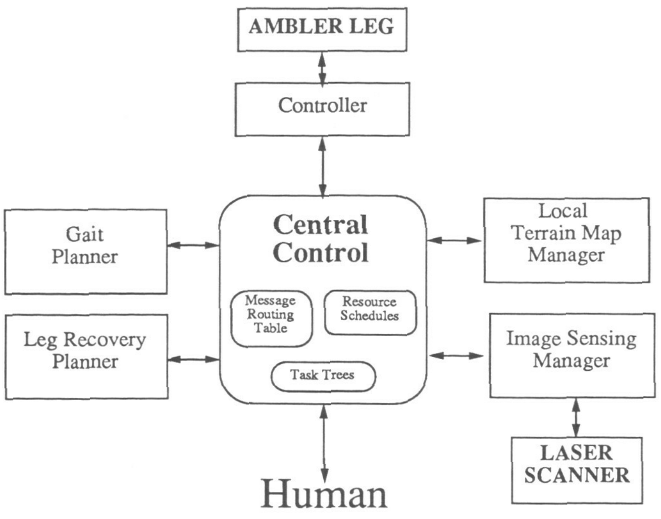


Figure 3: Modules for the Single-Leg Walking System

supports 1) distributed processing, 2) resource management, 3) hierarchical task decomposition, 4) temporal synchronization of tasks, 5) execution monitoring, and 6) error recovery.

The *Controller* module (Figure 3) handles all robot motions and responds to queries from other modules regarding leg position, carriage position and orientation, and force sensor readings. The Controller runs under the real-time vxWorks^{tm} operating system. The *Image Sensing Manager (ISM)* acquires scanner images from the Erim and determines the transformation from scanner to world coordinates. For debugging purposes, the ISM can also access images stored on disk. The *Local Terrain Map (LTM) Manager* processes scanner images to construct elevation maps of the terrain. The *Gait Planner* plans where to place the foot and how far to move the carriage in order to advance with minimal risk to the mechanism. The *Leg Recovery Planner (LRP)* determines a trajectory to the planned footfall location that is energy and time efficient and that avoids terrain collisions.

To walk the leg down the obstacle course, the user inputs a goal location along the rails. The walking system is totally autonomous from that point on. A message to plan (and execute) the walk is sent to the Gait Planner module. If the carriage position is close enough to the user-chosen goal, the Gait Planner signals success. Otherwise, it requests from the LTM Manager a terrain elevation map and a map that evaluates the potential support for the leg at various footfall locations. If the carriage position has changed from the last time a map request was issued, the LTM Manager requests a new scanner image from the ISM. In either case, the requested maps are constructed and sent back to the Gait Planner.

The Gait Planner combines constraints imposed by the terrain and footfall maps with geometric constraints on the leg's movement, and chooses the location that minimizes a weighted sum of the constraints. Based on the chosen footfall location, the Gait Planner chooses a body move that maximizes forward progress. The Gait Planner then sends the chosen footfall and body move to TCA, and then sends itself a message to plan the next step.

TCA forwards the footfall location to the LRP. The LRP uses a terrain map obtained from the LTM Manager to plan an obstacle-free trajectory. The trajectory is then forwarded through TCA to the Controller, which executes the trajectory and plants the foot at the desired location. After a successful leg move, TCA forwards to the Controller the body move generated by the Gait Planner. The Controller exerts enough force to compress the terrain, then relaxes to a force sufficient to provide traction. The horizontal (shoulder and elbow) joints are then actuated to drive the carriage forward. Finally, tension built up in the leg as a result of the body move is relieved, so that the leg does not slip when it is next lifted. At this point, the TCA forwards the message to the Gait Planner to plan out the next step.

Figure 4 presents a time breakdown by module for traversing a typical obstacle course. The system takes six steps in 13.5 minutes while covering about 8 meters (60cm/min). The darkly shaded areas of the chart represent times when a module is computing; lightly shaded areas are times when a module is awaiting a reply from another module. To reduce the chart's complexity, the 71 leg and body position queries to the Controller are not illustrated. In any event, they have a negligible effect on the timings since they are handled in less than 50msec each.

Figure 4 indicates that about 60% of the time is spent by the real-time Controller in moving the leg and carriage. Conversely the ISM, which spends only one half second for each of the seven images it acquires, is nearly always idle. Our measurements also show that the TCA central control module accounts for only about 3% of the total operating time. While, in theory, routing all messages through a central process could be a bottleneck, the evidence indicates that it is not a problem for this system.

We have used the walking system described above to navigate the single leg through a number of complex obstacle courses, such as illustrated in Figure 2. While not perfect (primarily due to sensor and mechanism inaccuracies), the system is generally successful at navigating the courses. The remainder of this paper describes perception, control and planning issues that we addressed in getting the system to walk on rugged terrain.

4 Issues in Perception

We use a scanning laser rangefinder because of the scanner's ability to directly recover the three-dimensional structure of the environment. Therefore, terrain maps can be constructed more rapidly and reliably than by passive vision techniques, such as binocular stereo or motion. In addition, using a laser scanner will enable the Ambler to walk at night. Although the scanner consumes more power than other imaging techniques, we believe its speed and accuracy more than offset this disadvantage.

Our primary terrain representation is an *elevation map*. An elevation map is a rectangular grid of real values, corresponding to the height of the terrain at a representative point within each grid cell (our current implementation uses the mid-point of the cell). Grid cells outside the scanner field of view are labeled *unknown*, and cells occluded by other objects are labeled as such, along with the maximum known elevation of the cell, given the available information. The map also contains an estimate of the uncertainty of the elevation value at each grid cell.

We chose to use elevation maps because 1) they provide a representation that is appropriate for a wide variety of tasks, 2) they can be constructed at multiple levels of resolution, 3) they are simple to manipulate, and 4) they can be accessed in a simple way (by a polygon that encloses the region of interest). A disadvantage is that our elevation maps record just a single value for each grid cell, hence overlapping objects (such as trees) cannot be represented. We do not, however, view this as a serious problem for navigating on Mars.

The LTM Manager uses the Locus Method to transform the raw range images into elevation and uncertainty maps [7, 5]. The Locus Method efficiently interpolates range data points to compute an evenly spaced grid of elevation points (Figure 5).

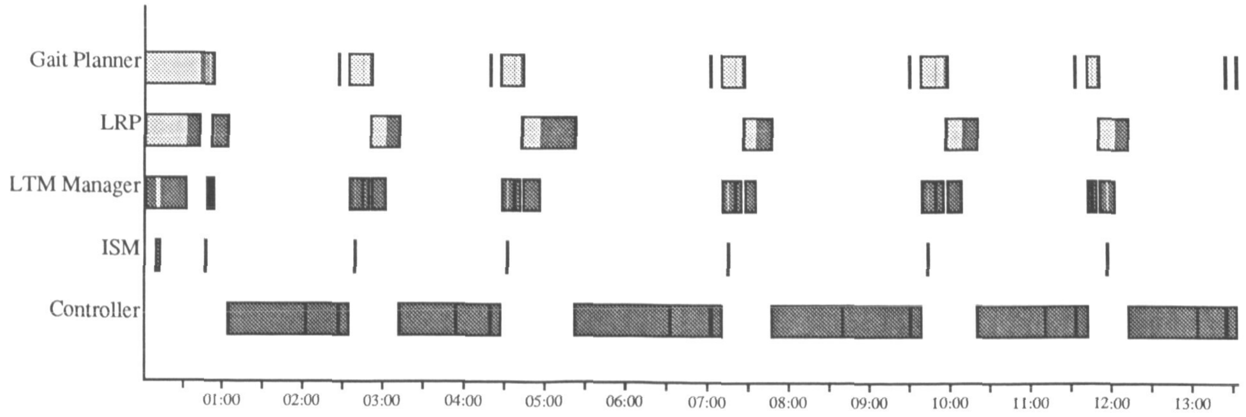


Figure 4: Timing Chart for a Typical Run

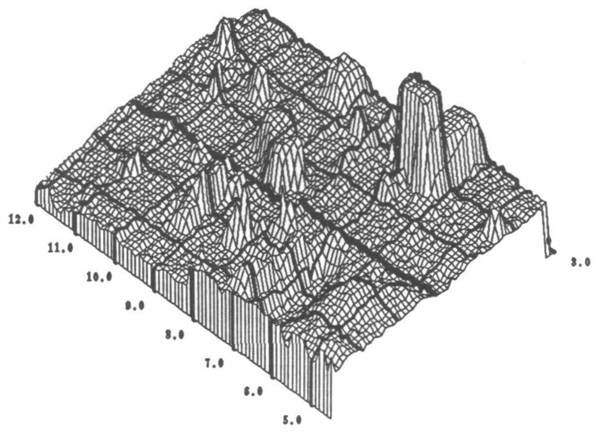


Figure 5: Elevation Map of Obstacle Course in Figure 2

The maps created from the most recent images are then merged with the current elevation maps using maximum likelihood estimation techniques. The merging operation is necessary because maps created from a single image do not, in general, have a wide enough field of view to support the necessary planning tasks. In planning a leg trajectory, for example, the LRP must take into account obstacles below and behind the vehicle. Because the scanner looks forward, the map constructed from the most recent range image cannot possibly cover this area, and so the planner needs a map constructed from a number of past images.

The portions of elevation maps requested by the planners are computed on demand, but are cached so that future queries that request the same (or overlapping) regions do not have to recalculate the values. Along with caching maps, we need a means of uncaching as the maps become larger than available local memory. The LTM Manager maintains a 20 by 20 meter window centered around the vehicle, outside of which grid cells are paged out, or clipped. While this method of computing on demand and caching is quite efficient, we are looking at pre-computing some maps concurrently with the planning and execution of walking commands.

5 Issues in Mechanism Control

The major issue we addressed in controlling the mechanism was the forceful interaction of the leg with the terrain. This impacted the leg and body move procedures, and also error detection and recovery performed by the real-time control system.

Moving the leg through free space posed few problems. The leg is moved through a series of user-supplied way-points, which are given in joint space. The Controller calculates the amount of time required for the slowest joint to move between successive way-points and then scales the speeds of the other joints so that all joints arrive at each way-point simultaneously. To smooth the motion, the way-points are linked with constant velocity segments connected together by constant acceleration segments [4].

For contacting the terrain, the motion command specifies that the last way-point is to be made in *transition mode.* In transition mode, the force/torque sensor is monitored and the motion is stopped if a specified (user-settable) force is achieved before the actual way-point is reached. If the way-point is reached first, a failure message is issued to TCA.

One problem encountered early in our experiments was the tendency of the leg to hit terrain features, even though obstacle-free paths were supposedly being followed. This was traced to inaccuracies in our kinematic model of the leg: we had initially assumed a rigid body, but the length of the leg and its method of connection to the rails led to a large amount of compliance in the mechanism. We partially solved this problem by measuring the deflections in the leg and updating the kinematic routines using a simple deflection model fit to the data. This improved the accuracy of the leg moves, as measured in Cartesian space, from about 20cm down to about 5cm.

More troublesome was the body move procedure. Our initial implementation commanded the position of the horizontal joints to follow a linear trajectory. This procedure proved to be very inaccurate due to the compliance of the mechanism, friction between the carriage and rails, and compliance of the terrain. We often witnessed errors of more than 40cm over a (commanded) one meter body move.

Our remedy was to use a velocity, rather than position, control procedure. To move the body, the force on the leg is first increased to 800 pounds, to compress the underlying terrain. The force is then relieved to 500 pounds, which provides sufficient tractive force. The shoulder and elbow joints are then commanded to achieve given velocities. First, the Cartesian velocity of the carriage is computed as a clipped, linear function of the error between the present carriage position (as read from the potentiometer) and the commanded goal position. This velocity is then converted into joint velocities using an inverse Jacobian function. The body move control loop is operated at a frequency of about 60 Hz, which differs sufficiently from the natural frequency of the system so that resonance does not occur.

This velocity-controlled body move procedure is accurate to within 5cm. The algorithm was subjected to extensive testing to gain confidence in its performance. Over 1000 moves were performed with the leg starting at various X, Y locations relative to the carriage. The resultant data not only confirmed the general accuracy of the body move procedure, but also provided a ''cost map'' for the Gait Planner to indicate how far the carriage can reliably advance from different footfall locations (see Section 6).

During a body move, the compliance of the mechanism causes overshoot of the expected positions of the joints, assuming a rigid kinematic description of the leg. This overshoot takes the form of stored strain energy which causes the foot to drag across the terrain when the leg is next lifted. To prevent this, the tension is relieved by adjusting the final joint angles to correspond with the expected Cartesian position of the leg.

The control system also contains several procedures for detecting and reacting to errors. The joint limit sensors and the motion control cards are continually monitored for possible failures. During the body move control loop, the system monitors the forces exerted on the foot. The leg is stopped if the force drops off rapidly, indicating that the foot may have broken free. As described above, the force sensor is also monitored during transition mode to detect when the terrain is contacted.

When errors are detected, the Controller halts any ongoing leg motions and informs TCA, which passes the failure message on to the appropriate exception handler. In addition, the control software permits recovery from hardware errors without restarting the entire walking system. Such errors include tripping limit switches, amplifier faults, servo errors, excessive force readings, and ''kill'' messages from users.

6 Issues in Planning

Planning problems for single-leg walking include deciding where to plant the leg, how to move it through space, and how far to move the carriage at each step. Our approach utilizes constraints imposed by the robot's design to plan movements that are efficient, reliable, and provide a good rate of progress for the mechanism.

The Gait Planner plans footfalls by combining various geometric and terrain constraints. For each constraint, a *cost map* is created that indicates the goodness of the constraint within each grid cell (Figure 6). The cost maps are combined using a weighted sum, and the grid cell with the lowest cost is chosen as the footfall location. The Gait Planner then chooses a body move that is the minimum of 1) the best possible advance from the chosen location, and 2) a user-defined threshold (we typically constrain the body advance to 1.5m to get a reasonable number of footfalls over the length of our testbed).

The constraints used by the Gait Planner were derived from both analysis and experimental evidence. The geometric constraints include 1) the mechanical limits of the leg, 2) how far the carriage can travel from a given footfall location, which is based on empirical values derived from testing the controller's body move algorithm (Section 5), and 3) the visibility of the leg in the scanner field of view, to avoid occluding terrain. Terrain constraints include 4) the terrain elevation, since the leg cannot reach areas that are too high or too low, 5) an evaluation of the flatness of the terrain around each grid cell [3], since relatively flat terrain is preferable both for stability and for providing traction in moving the body, and 6) the closeness of the footfall location to adjacent obstacles, in order to compensate for inaccuracies in the mechanism control and scanner resolution.

In combining the cost maps, constraints 1 and 4 above are used as binary constraints: if the location is not reachable, it is eliminated from consideration, no matter what the other values are. The remaining two terrain constraints are given high weights relative to the remaining two geometric constraints. This reflects our concern for the safety of the machine over its progress.

Advantages of this constraint-based approach are that 1) the planner does not have to commit *a priori* to which constraint is most important, and 2) it is easy to add new constraints as relevant ones are identified [12]. Although this approach evaluates a large number of grid cells, in practice the gait planning is fast relative to other computations.

Once the Gait Planner decides where to put the foot, the LRP determines the trajectory that will get the leg to that position without hitting any obstacles. The LRP uses the novel *Envelope Trajectory Finding Algorithm (ETFA)* to find time and energy efficient moves through 3D space, while searching only a 2D grid. The LRP starts by creating a configuration search space for the elbow and shoulder joints [8], dividing the

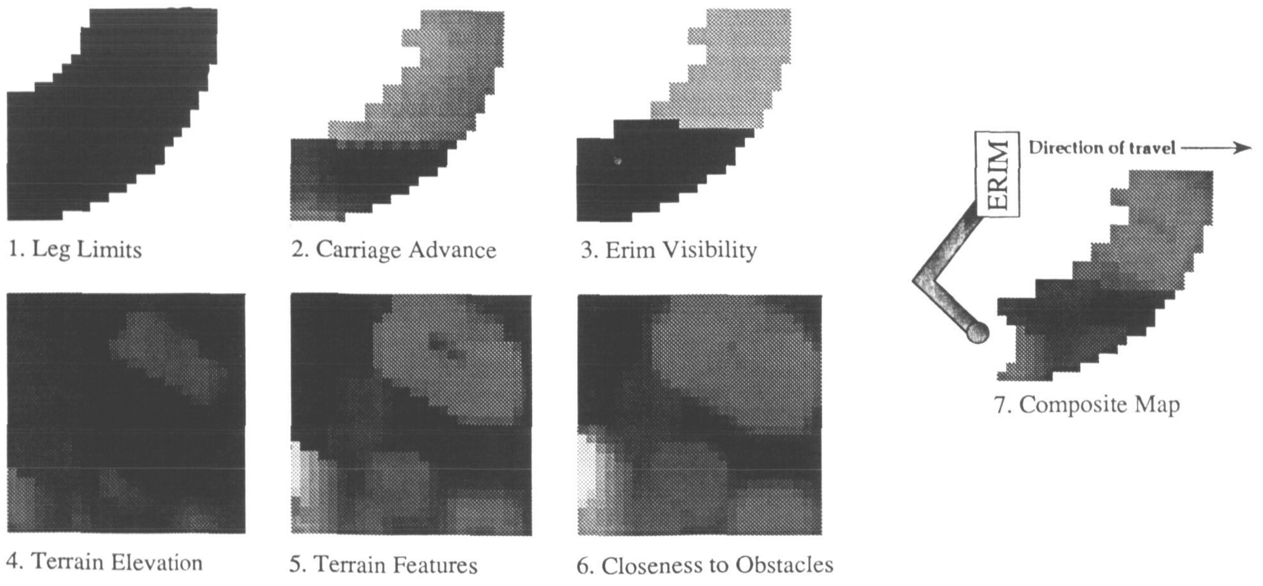
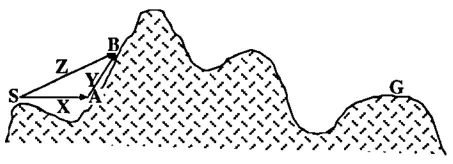


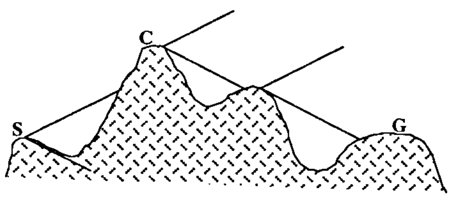
Figure 6: Constraint Cost Maps for Choosing a Good Footfall (darker shades indicate better footfall locations)

space into a discrete grid approximately 0.1 radian wide. The LRP fills the grid with obstacles, growing the terrain features and other legs (for the six-legged case) by the radius of the foot plus an uncertainty factor.

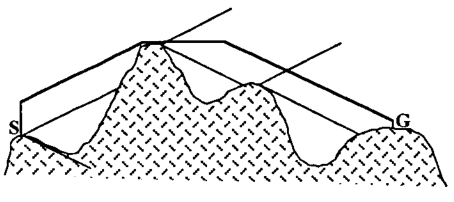
To search the space, the ETFA needs to estimate the energy and time needed to travel between grid cells. The energy consumed is estimated simply as the sum of the energy needed to move the elbow and shoulder joints to a cell, plus the energy needed to raise the leg above the terrain elevation at the cell. While this assumes that the power consumption in each joint is independent, it is a reasonable approximation given the slow speeds of our mechanism.



a. Different Trajectories



b. Trajectory Envelopes



c. Final Path

Figure 7: The Envelope Trajectory Finding Algorithm

It is more difficult to estimate the time needed to get to a cell, as Figure 7a illustrates. If we just add up the times to get to each individual cell, path **X** is the quickest way to get to point **A**. To get a little further to point **B**, however, path **Z** is faster than **X** followed by **Y**, since in path **Y** the horizontal joints must stop and wait for the vertical lift, while in path **Z**, the leg is lifting while it is moving horizontally.

In essence, we need to keep track of all possible paths that the leg can take in reaching a particular grid cell. This is what the "envelope" part of the ETFA is about. The algorithm keeps track of the maximum and minimum heights that the leg can reach in any particular cell, assuming that the leg lifts/lowers at full speed while moving horizontally (Figure 7b). Thus, the leg can reach anywhere within the envelope in the same amount of time. Only if the terrain is above the top of an envelope (e.g., point **C**) does the leg have to stop moving horizontally and lift.

The ETFA finds the minimum-cost trajectory using A* search and a weighted sum of the energy and time metrics described above. At the end of the search, the planner determines an actual trajectory through the envelope space by choosing vertical moves that minimize the risk to the machine while maintaining the optimality of the path found. In particular, this means performing all purely vertical lifts at the start and delaying all purely vertical descents until the end of the move (Figure 7c).

In actual use, the Gait Planner performs very well, typically choosing safe footfalls that skirt obstacles, while enabling the carriage to be moved at, or near, its maximum advance. The LRP typically chooses trajectories that hug the ground when the terrain is relatively flat. For obstacle-filled terrain, the LRP typically chooses to go around, rather than over, large obstacles, since the vertical joint of the leg is much slower than the two horizontal joints.

7 Conclusions

To date, the leg has autonomously traversed several hundred meters through various obstacle courses. The effort has taught us much about perception, locomotion, and planning for rugged-terrain walking, lessons that apply to the full six-legged Ambler.

Perhaps the most important result is that our experience with the single-leg testbed has led to some significant changes in the configuration of the Ambler, especially with regard to compliance. The single leg was too flexible to permit the type of accurate control needed to negotiate very rugged terrain. The legs of the new Ambler design are extremely rigid [1]. Our experience to date with the full Ambler indicates that we can do leg and body moves to within a centimeter of commanded positions. In any event, we believe our experience with the single-leg testbed will enable us to handle any residual compliance.

As for the software system, the Task Control Architecture has been ported to the Ambler without any modifications. The LTM Manager and ISM needed only minor modifications to handle the new Ambler geometry. The Erim scanner itself, however, was found to have insufficient resolution and accuracy for our purposes. While this did not prevent successful walking, it did limit the roughness of the terrains that the system could traverse. For the six-legged Ambler we have procured a scanner, manufactured by Perceptron, that overcomes most of these problems.

One surprise in the endeavor was the fine balance between geometric and terrain constraints for gait planning. Much of our effort in getting the leg to negotiate terrain was in fine-tuning the weighting function that combined constraints. Our current methodology is empirical: trying the system on a variety of terrains and tweaking the weights to reflect the results of the experiments. To make the process of choosing weights less *ad hoc*, we are considering the use of adaptive algorithms that autonomously adjust the constraint weights based on the difference between the planned moves and actual outcomes. Another problem was that the footfall evaluation constraints used did not always yield what we subjectively believed to be the best footfall location. We are currently investigating a more feature-based approach to provide better evaluations. In general, gait and footfall planning are areas of on-going research and will undoubtably consume much of our effort in getting the Ambler to walk on rugged terrain [13]. We believe, however, that the constraint-based structure of the Gait Planner will enable us to experiment with various constraints and weighting schemes without much alteration to the basic planning algorithm.

Error detection and recovery is an important area that, to date, has received only modest attention by our group. The real-time Controller continually monitors its sensors and electronics to detect anomalies, and halts the mechanism when they occur. It then passes error information through TCA for action by higher-level exception handlers. Currently, the exception handlers halt the system if the error was caused by a hardware fault (e.g., a bad amplifier), and replan the last step if the error was caused by a bad footfall (e.g., the foot slips while doing a body move). Much more work remains, however, in detecting additional errors (such as colliding with obstacles while moving through space), automated diagnosis of errors, and intelligent error recovery.

The major impetus for the single-leg walking program was to gain experience for six-legged walking. To that extent, the project was quite successful. We have gained much insight into perceiving and modeling rugged terrain, controlling the legged mechanism, interacting with the ground, choosing safe yet effective footfalls, and planning efficient leg moves through space. The task ahead is to apply our experiences and successes to an autonomous walking system for the full six-legged Ambler.

Acknowlegements

Many members of the Planetary Rover project contributed to the single-leg walking system. Significant contributions in the integration and testing of the system were provided by P. Balakumar, L. Chrisman, C. Fedor, R. Hoffman, M. Hebert, G. Roston, and D. Wettergreen. This research is supported by NASA under Contract NAGW-1175.

References

1. Bares, J., Whittaker, W. Walking Robot with a Circulating Gait. Proc. of IEEE International Workshop on Intelligent Robots and Systems, Tsuchiura, Japan, July, 1990.

2. Bares, J., et al. Ambler: An Autonomous Rover for Planetary Exploration. IEEE Computer, Vol. 22, No. 6, 1989.

3. Caillas, C.,Hebert, M.,Krotkov, E., Kweon, I.S., and Kanade, T. Methods for Identifying Footfall Positions for a Legged Robot. Proc. IEEE International Workshop on Intelligent Robots and Systems, Tsukuba, Japan, September, 1989, pp. 244-250.

4. Craig, J.. *Introduction to Robotics, Mechanics and Control.* Addison-Wesley Publishing Company, 1986.

5. Krotkov, E.,Caillas, C., Hebert, M., Kweon, I.S., and Kanade, T. First Results in Terrain Mapping for a Roving Planetary Explorer. Proc. NASA Conf. on Space Telerobotics, Jet Propulsion Laboratory, Pasadena, CA, January, 1989.

6. Krotkov E., Simmons, R., Thorpe, C. Single-Leg Walking with Integrated Perception, Planning, and Control. Proc. of IEEE International Workshop on Intelligent Robots and Systems, Tsuchiura, Japan, July, 1990.

7. Kweon, I. S. and Hebert, M. and Kanade, T. Perception for Rugged Terrain. Proc. SPIE Mobile Robots III Conf., Cambridge, Massachusetts, November, 1988.

8. Lozano-Perez, T. Spatial Planning: A Configuration Space Approach. IEEE Transactions on Computers, C-32:108-120, 1983.

9. Simmons, R., Mitchell, T. A Task Control Architecture for Autonomous Robots. Proceedings of Space Operations and Autonomous Robotics Conference, Houston, TX, July, 1989.

10. Simmons, R., Krotkov, E., Roston, G. Integrated System for Single Leg Walking. Tech. Rept. CMU-RI-90, Robotics Institute, Carnegie Mellon University, July, 1990.

11. Simmons, R., Lin, L.J., Fedor, C. Autonomous Task Control for Mobile Robots. Proc. of IEEE Symposium on Intelligent Control, Philadelphia, PA, September, 1990.

12. Stentz, A. Multiresolution Constraint Modeling for Mobile Robot Planning. Proc. SPIE Symposium on Advances in Intelligent Robotics Systems, November, 1989.

13. Wettergreen, D., Thomas, H., and Thorpe, C. Planning Strategies for the Ambler Walking Robot. Proc. IEEE International Conference on Systems Engineering, August, 1990.

SENSING AND PERCEPTION

N91-20655

COMPARISON OF FORCE AND TACTILE FEEDBACK FOR GRASP FORCE CONTROL IN TELEMANIPULATION

Steven F. Wiker, Assistant Professor
Neil Duffie, Associate Professor
Thomas Y. Yen, Research Assistant
Karen Gale, Research Assistant

Wisconsin Center for Space Automation and Robotics
University of Wisconsin, Madison,WI, 53706, USA

ABSTRACT

This study examines the comparative efficacy of using direct force feedback or a simple vibrotactile display to convey changes in the intensity of remote grasp force relayed from a robotic end effector. Our findings show that a simple vibrotactile cue, in absence of direct force feedback, is effective in signalling abrupt changes in remote grasp force regardless of magnitude, and when changes in force are not too slow or protracted in nature (i.e., ramp times less than 2 s). In cases where the operator must dynamically tract and respond to slow but large variations in grasp force, the comparatively crude vibrotactile display examined in this study would prove helpful; but would not be as effective as that of a direct contact force display. Immediate applications and utility of current generation and near-term prototype tactile displays are discussed.

INTRODUCTION

Many remote manipulators provide only visual feedback to guide remote end-effector pose and grasp force. As a result, operators can command insufficient grasp force to the remote controller. Consequences of inadequate grasp force are: a) slippage and realignment of objects held within the remote gripper, b) complete loss of grasp, and c) increased risk of task or mission failure. Given such consequences, operators usually apply greater than necessary grasp force to the master-controller following a better safe, than sorry strategy for control of remote grasp force.

Unfortunately, sustained or very repetitious overforcing of the master controller can be counterproductive if applied forces are sufficient to:

a) damage to objects held within the remote gripper,

b) provoke localized muscle fatigue and discomfort (Wiker, Hershkowitz, and Zik (1989), see Wiker, Chaffin and Langolf (1989) for bibliography), and

c) promote degradation of manual performance (see Wiker, Langolf, and Chaffin (1989) for bibliography).

A frequently advocated solution for such problems is to provide bilateral, force-reflection between the master controller an remote end-effector. Once equipped, such telemanipulators typically demonstrate much improved manipulative performance. Provision of force reflection is not, however, without its price. Bilateral force reflection:

a) is usually quite expensive to build and then to maintain, and

b) nearly precludes post hoc implementation with existing telemanipulators.

In comparison with force feedback, current generation tactile displays:

a) are usually inexpensive to build, implement, and to maintain,

b) like force reflective displays, can provide sensory information that is consistent with that normally experienced during typical manual activities. Hence, the operator does not have to create novel perceptual models that require constant reinforcement, and

c) can be combined with existing telemanipulators to augment visual feedback to enhance and to extend operation manipulative capabilities.

For these reasons, we were interested in the efficacy of augmenting a telemanipulator with only tactile or vibrotactile cues of grasp force (Wiker, 1988a, b, and c). Specifically, we were interested in how effectively an operator could use either a direct force feedback or cutaneous cue to detect changes in displayed remote grasp force, and to regain desired levels of grasp force.

METHODS AND MATERIALS

Subjects

Seven male and two female university students participated in this experiment on a voluntary, paid, and informed consent basis. All subjects appeared and claimed to be in good health.

Apparatus

An electromechanical, one degree-of-freedom, bilateral, master-slave telemanipulator, was shown in Figure 1, was used this experiment. A microcomputer was used to monitor and actuate direct-drive electric actuators that produced negligible friction and backlash (See Duffie, Wiker, and Zik (1989) and Duffie, Wiker, Zik, and Gale (1990) for a more detailed explanations of the master-slave apparatus employed in this experiment).

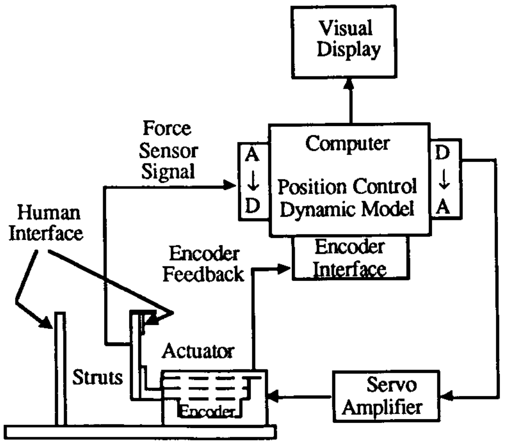


Figure 1.

Diagram of the master-controller system used in the experiment.

A calibrated strain gage was mounted on the master controller to measure forces exerted by the subject's fingers when squeezing the master controller digits. A high-resolution encoder monitored the actual position and velocity of the controller's digits. The microcomputer was used to command and to monitor the position of the master-controller, and to record:

a) position commands sent to the master-controller,

b) actual position of the master-controller,

c) force or vibrotactile intensity presented to the subject, and

d) time.

Force cues were produced by monitoring commanded and actual position, and converting the position error into a reactive force using a spring constant of 0.833 N per mm.

To provide vibration stimuli to the subject, the master controller actuator was oscillated at 250 Hz. Cues of changes in remote force were signalled by altering peak-to-peak amplitudes of master controller digit oscillations. To avoid the difficulties of mechanical couplings, we maintained contact between the master controller digits and subject's fingers using a servo-controlled contact force of 1.43 N. This strategy helped to stabilize the mechanical impedance of the finger tissues and reduced the potential for variable mechanical damping of the vibration stimulus.

Perceived intensities of the force and vibrotactile cues were matched for each individual subject using a cross-modal matching technique (See Lodge (1981) and Wiker et al. (1989) for detailed procedures). Thus, a change in remote grasp produced a change in master-controller force reflection, or in vibrotactile vibration intensities, that were perceived to be of equal intensity.

PROCEDURES

Subjects performed a series of trials in which they maintained a pulp-pinch grasp of fixed force magnitude at a "remote gripper." The magnitude of the remote grasp force was fixed for each subject based upon their psychophysical estimate of 5 N. The average grasp force produced across all subjects was 6.2 N. Grasp force applied by the subjects was indicated by a corresponding adjustment of a visual cursor position on a CRT. Subjects exerted and maintained the required pinch grasp until they felt confident that they could recognize and correct any changes in the level of grasp force held without the aid of visual feedback.

Once subjects had signalled to eliminate the visual indicator of grasp force, a random time interval ranging between 2 and 5 s passed before the computer moved the digits of the master-controller either 2, 4, or 6 mm away from the subject's finger. The maximum displacement (6 mm) produced a reduction in force or vibration cue without significant change in the hand's posture. Subjects were instructed to use force or vibrotactile cues, depending upon the trial, to detect a change in remote grasp force, and to initiate and guide adjustments in grasp posture required to return grasp forces back to the objective force as quickly and as accurately as possible.

Once the disturbance in force or vibrotactile cue was initiated, adjustments in the master controller digit position and actual grasp force or vibrotactile intensity were recorded at 166.7 Hz until completion of a 6 s post-disturbance period.

Experimental Paradigm

As shown in Figure 2, subjects performed a series of trials with, either force or vibrotactile feedback, in which we changed the magnitude of the grasp force disturbance (i.e., change in master controller position of 2, 4, or 6 mm), and the rate at which the disturbance was invoked (a step change, a 2 s linear ramp, or a 4 s linear ramp). All nine combinations of positional displacement and displacement rate were presented five times, in random order, using either force or vibrotactile feedback, during a single 1 hour period. Subjects received, on average, one minute rest intervals between trials. Trials performed using the alternative display mode were completed within a few days of the initial day's testing. The order of experience of grasp force display format was randomly assigned.

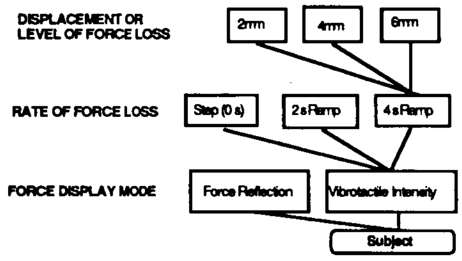


Figure 2.

Levels and rate of change in grasp force displayed to each subject using a repeated measures experimental paradigm.

Graphically displayed in Figure 3, metrics used to characterize subject grasp control capability were:

a) maximum loss of grasp force, or force error, following grasp disturbance,

b) time intervals needed by subjects to return grasp forces to within 90 percent of the pre-disturbance grasp force magnitude (i.e., the grasp force recovery period), and

c) difference between pre- and post-disturbance grasp force during the last 2 s of 6 s recovery period.

Ideal performance would be characterized by no grasp force error during the disturbance period. If some force loss was experienced, then a subject should rapidly reestablish the desired force level with no differences in grasp forces measured during pre- and during the final stages of post-disturbance force recovery period.

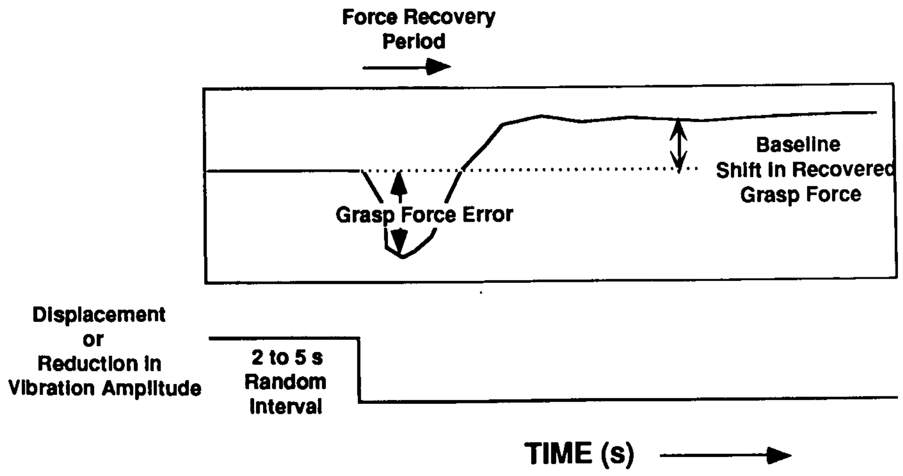

Force Recovery Period
Baseline Shift In Recovered Grasp Force
Grasp Force Error
Displacement or Reduction In Vibration Amplitude
2 to 5 s Random Interval
TIME (s)

Figure 3.
Diagram showing subject performance criteria used to evaluate their force control capacity following an unexpected reduction in remote gripper grasp force indicated by a change in the intensity of contact force or vibration amplitude at the master-controller's digits.

RESULTS

Remote Grasp Force Display Mode and Initial Grasp Force Error

A repeated-measures ANOVA was performed to test whether remote grasp force feedback display mode, the magnitude of the grasp force disturbance, the rate at which the disturbance occurred, and their interactions were important determinants in the control of remote grasp force. All tests were conducted fixing Type I and Type II errors at p=0.05 and p=0.10 respectively. The mode of force display, the magnitude of shift in force (i.e. controller displacement), the rate at which changes in force cues occurred (i.e., period of the displacement ramp), as well as all two- and three-way interactions of these factors, showed statistically significant impacts upon operator grasp force control ($p < .05$; see ANOVA tables in Appendix for F-tests).

As shown in Figure 4, regardless of feedback mode, subjects were unable to maintain grasp force control with zero error following either a step or ramp change in master controller force feedback. Loss of grasp force control was directly proportional to the speed at which the master controller indicated that the "remote" grasp force had declined. On average, the force display was more effective in minimizing grasp force error following a decline in feedback intensity. However, vibrotactile feedback produced equivalent performance with that of the force display when disturbances were rapid (i.e., a step-reduction in vibration intensity), and when shifts in intensity of grasp force cues were small. The vibrotactile display was inferior in maximizing grasp control when changes in force remote intensity were quite slow and protracted in nature.

Remote Grasp Force Display Mode and Force Recovery Period

In addition to reduction of the maximum loss of gripper force, an effective display should help the operator to quickly regain desired grasp force once lost. Our analysis of the period of time required for subjects to regain 90 percent of initial force levels following a step or ramp loss of force, showed that:

a) for small reductions in grasp force, the amount of time required to increase force to 90% of the original force were equivalent between vibrotactile and force reflective displays. However, as the magnitude of grasp force change increased, direct force feedback improved performance while vibrotactile cues were associated with longer recovery periods,

b) use of vibrotactile display of remote grasp force produced an opposite effect from that observed with direct force reflection. Recovery was more rapid when vibrotactile stimulus changes were small (i.e., in the face of small manipulator displacements from the finger).

Remote Grasp Force Display Mode and Error in Recovered Grasp Force

Another metric of the subject's ability to control grasp force is the error between the pre- and post-disturbance level of grasp force. About 70 percent of the trials produced under-force errors. If displacement or loss of force was small (i.e., 1.7 N), then vibrotactile displays produced the most accurate return to desired grasp force. However, as the magnitude of force disturbances increased, use of the vibrotactile display produced a progressively lower levels of recovered grasp force and, thus, greater errors in recovered grasp force. This outcome was exacerbated when rates of changes in displayed force, or lengths of time subjects had to spend tracking changes in grasp force, were increased. All remaining effects were not found to be statistically significant.

Relationships Found Among Force Display Modes and Grasp Force Disturbance Parameters

Correlation analysis was performed among dependent metrics of grasp force as well as independent factors such as force display mode, magnitude of force loss or manipulator displacement, and rate of force loss or change in manipulator displacement. The analyses showed the following material relationships:

a) grasp force error was directly associated with the rate at which the loss of grasp force occurred ($r= -0.78$ for force display and $r = -0.55$ for vibrotactile display),

b) the magnitude and direction of error in recovered grasp force was directly related to the magnitude of the initial loss of grasp force when using the vibrotactile display ($r = -0.61$),

c) differences between pre- and post-disturbance grasp force were lower when subjects spent more time establishing the desired grasp force ($r = -0.78$ for vibrotactile displays, $r = -0.66$ for force displays),

d) maximum loss in grasp force was greatest when ramp periods were small or when the rate of change in displayed force was high ($r = -0.40$ vibrotactile, $r = -0.65$ for force display)

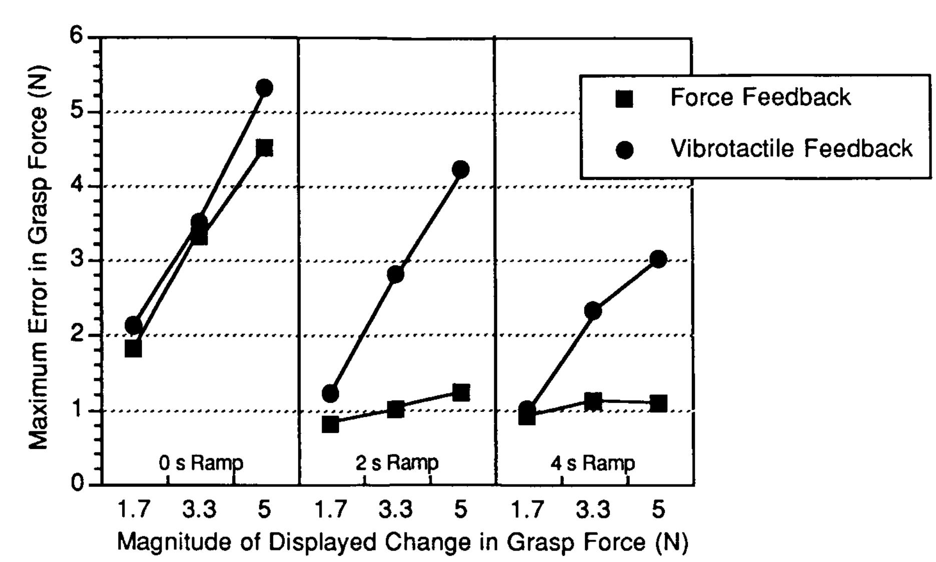


Figure 4.

Maximum error in remote grasp force following a change in contact force or vibration intensity displayed at the master controller. Errors are plotted against plotted across display mode, and magnitude and rate of change in force displayed .

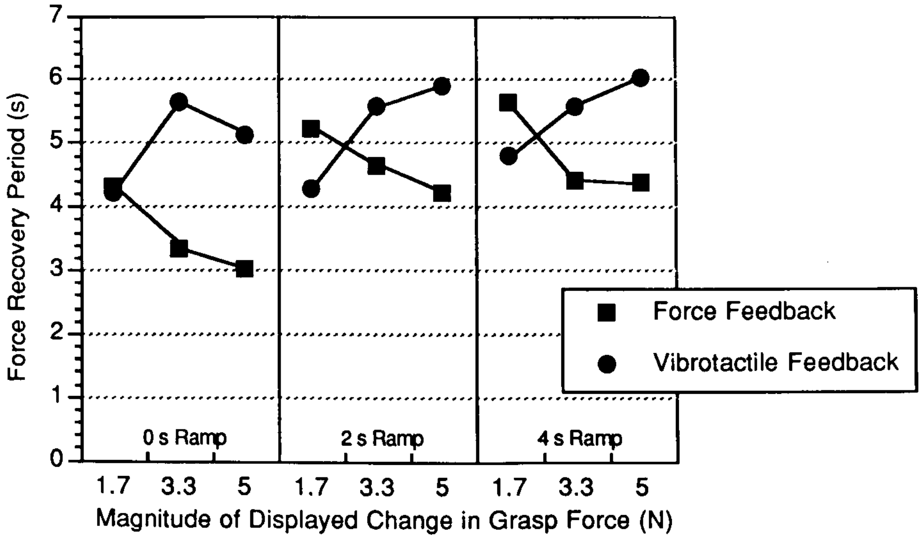


Figure 5.

Time period required to reestablish 90 percent of the pre-disturbance level of grasp force plotted across display mode, and magnitude and rate of change in force displayed.

Predictions of Grasp Force Response Error

$$GFE = 0.55 - 0.16\,D + 0.42\,F + 0.33R + 0.30\,DF - 0.20\,DR - 0.30\,FR + 0.10\,DFR$$

$$R^2 = 0.71$$

where:

GFE	=	Magnitude of Maximum Difference Between Pre and Post-disturbance Grasp Force
D	=	Display Mode (0 = force, 1 = vibrotactile)
F	=	Magnitude of Displayed Change in Grasp Force (N)
R	=	Duration of Ramped Change in Displayed Grasp Force (s)

The interrelationships between the magnitude and rate of change of displayed shifts in remote grasp force and error in commanded grasp force predicted by the above equation are summarized in the following response surfaces plotted for each display mode:

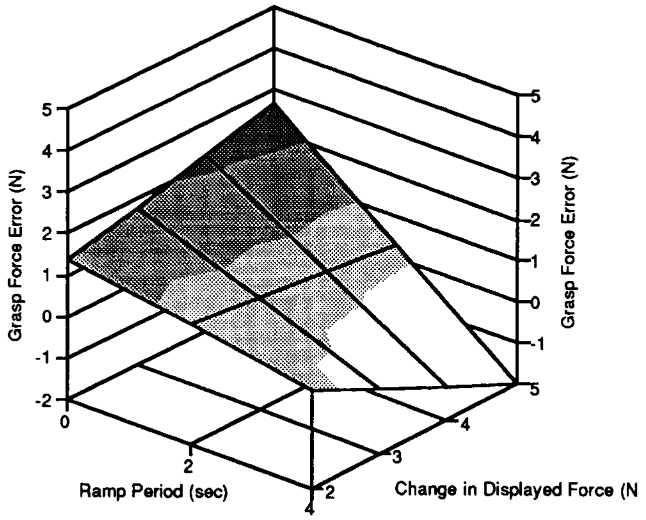


Figure 7.
Predicted grasp force error produced by direct force feedback of changes in remote grasp force.

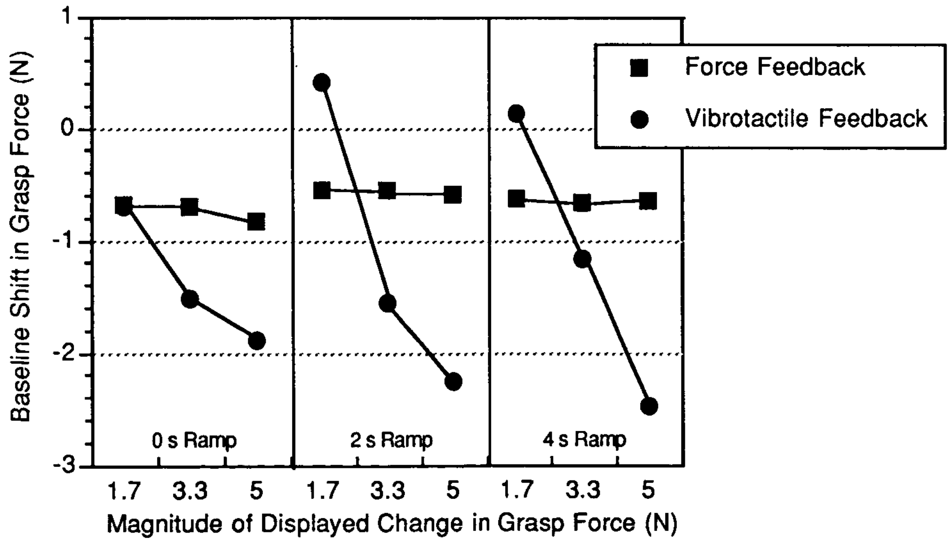


Figure 6.

Error in recovered grasp force plotted across display mode, and magnitude and rate of change in force displayed.

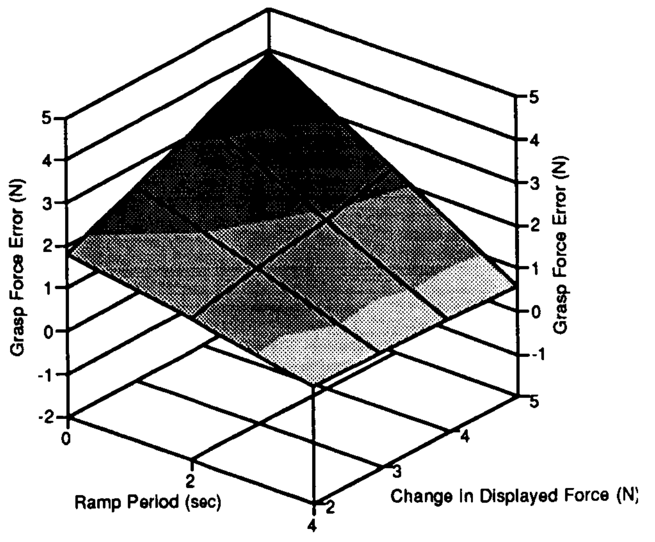


Figure 8.
Predicted grasp force error produced by vibrotactile feedback of changes in remote grasp force.

DISCUSSION AND CONCLUSIONS

Our findings show that a vibrotactile display can signal changes in and guide control of remote grasp force. A comparatively simple vibrotactile display showed equivalent or improved performance with that of a force reflective display when changes in displayed remote grasp force were abrupt (i.e., step changes) or when magnitudes of shifts, regardless of ramp period up to 4 s, were small. However, when changes in remote grasp force were larger in magnitude, and required sustained adjustment of the position of the master controller, the direct display of force produced better control of remote grasp force.

With extremely abrupt or step-like changes in remote grasp force, perceptual and motor response delays produced decrements in grasp force that were directly proportional to the magnitude of the disturbance; regardless of display mode used. In short, subjects were able to recognize and respond to the displayed disturbance with equal capability across display modes. Given a fixed response delay due to basic neuromotor reaction time requirements, the errors found were proportional to the magnitude of the abrupt change in grasp force.

If force adjustments were greater in magnitude or more sustained in nature, performance using the vibrotactile display was worse than that found with display of direct contact force. This outcome may be due to one or all of the following:

a) greater delays in processing changes in vibratory cutaneous stimuli in comparison to those found with force perception, (light touch transition to muscle tension sense),

b) efferent masking of cutaneous feedback as the subject's digits continued to adjust the position of the master controller's digits,

c) masking of small changes in the vibratory stimulus by the stimulus itself.

The ever-present and tenacious phenomena of efferent and afferent masking of cutaneous stimuli have been reported in the literature. Although further basic research is needed to fully characterize the nature and magnitude of masking effects, such effects can be mitigated to some degree. We expect that future experiments will show that changes in tactile display locus and changing both the intensity and spatial organization of the stimulus representing grasp force intensity and distribution will produce displays that are far more competitive with direct force reflection displays that the simple system investigated here.

Our findings show that a simple vibrotactile cue, in absence of direct force feedback, can be very effective in signalling abrupt changes in remote grasp force regardless of magnitude, and when changes in force are not too slow or protracted in nature (i.e., ramp times less than 2 s). For a large variety of remote manipulation tasks, force cues needed would not be expected to exceed those examined in this experiment. If so, vibrotactile or similar forms of tactile displays would be effective in aiding remote grasp and manipulation. In cases where the operator must dynamically tract and respond to slow but large variations in grasp force, the vibrotactile display examined in this study would still prove helpful; but not as effective as that of a contact force display.

We are pursuing development of tactile displays that are more comfortable to use for long periods of time (i.e., between 1 and 2 hours), that provide patterns of cutaneous cues that are more resistant to masking effects, and that can provide cues of variations in magnitude and direction of forces distributed across the remote contact surfaces. Current generation and near-term prototype tactile displays under development by WCSAR industrial consortia members will provide additional sensory information needed by operators of visually remote manipulators that cannot practically employ high-quality bilateral direct force feedback, to wearers of prosthetic limbs, and to operators of telemanipulator systems in microgravity environments where applying forces to the operator's body becomes problematic.

REFERENCES

Duffie, N.A., Wiker, S. F. and Zik, J. J. (1989) Test bed experiments for various telerobotic system characteristics and configurations. Presented at the Annual Space Operations Automation and Robotics Conference in Houston, TX , July 25-27th.

Duffie, N.A., Wiker, S. F., Zik, J. J., and Gale, K. L. (1990) Impact of inertia, friction and backlash upon force control in telemanipulaton. Presented at the Annual Space Operations Automation and Robotics Conference in Albuquerque, NM , July 23-26 th.

Lodge, M. (1981) **Magnitude Scaling: Quantitative Measurement of Opinions**. London: Sage Publications.

Wiker, S. F. (1988a) Tactile sensing capabilities for telerobotics. Naval Ocean Systems Center,Technical Report No. 1249.

Wiker, S. F. (1988b) Tactile sensing techniques for robots. In E. Heer and H. Lum (Eds.) **Machine Intelligence and Autonomy for Aerospace Systems** Washington, DC: American Institute of Aeronautics & Astronautics Press.

Wiker, S. F. (1988c) Teletouch display development. Naval Ocean Systems Center Technical Report No. 1230, San Diego, CA.

Wiker, S. F. (1990) Telerobotic Design Issues. Annual Meeting of the Industrial Engineering Society, San Francisco, CA.

Wiker, S. F. , Hershkowitz, E., and Zik, J. (1989) Teleoperator comfort and psychometric stability: Criteria for limiting master-controller forces of operation and feedback during telemanipulation. Presented at the National Aeronautics & Space Administration's Conference on Space Telerobotics, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA.

Wiker, S. F., Chaffin, D. B., and Langolf, G. D. (1989) Shoulder posture and localized muscle fatigue and discomfort. **Ergonomics** 32(20):211-237.

Wiker, S. F., Langolf, G. D., and Chaffin, D. B. (1989) Arm posture and human movement capability. **Human Factors** 31(4): 421-442.

ACKNOWLEDGEMENTS

This work was supported by the Wisconsin Center for Space Automation and Robotics in part by NASA under grant NAG-W975 and by a consortium of industrial sponsors.

APPENDIX

ANOVA Table for Maximum Force Error

Source of Variation	df	Sum of Squares	Mean Square	F	p <
Subjects	8	16.787	2.098		
Feedback Mode (M)	**1**	**49.093**	**49.093**	**22.3**	**.0015**
Error	8	17.603	2.200		
Force Magnitude (F)	**2**	**98.612**	**49.306**	**155.8**	**.0000**
Error	16	5.065	.317		
MF	**2**	**18.551**	**9.276**	**23.2**	**.0000**
Error	16	6.388	.399		
Ramp Speed (R)	**2**	**102.197**	**51.099**	**71.95**	**.0000**
Error	16	11.362	.710		
MR	**2**	**10.773**	**5.387**	**16.64**	**.0001**
Error	16	5.180	.324		
FR	**4**	**17.540**	**4.385**	**11.55**	**.0000**
Error	32	12.145	.380		
MFR	**4**	**5.750**	**1.437**	**4.39**	**.0061**
Error	32	10.484	.328		

ANOVA Table for Force Recovery Period

Source of Variation	df	Sum of Squares	Mean Square	F	p <
Subjects	8	100.163	12.520		
Feedback Mode (M)	1	31.210	31.210	1.51	.2534
Error	8	164.818	20.602		
Force Magnitude (F)	2	.380	.190	.06	.9349
Error	16	44.920	2.807		
MF	2	47.108	23.554	14.3	.0003
Error	16	26.239	1.640		
Ramp Speed (R)	**2**	**21.881**	**10.941**	**4.87**	**.0222**
Error	16	35.899	2.244		
MR	2	5.515	2.758	2.17	.1468
Error	16	20.346	1.272		
FR	4	3.047	.762	.65	.6295
Error	32	37.368	1.168		
MFR	4	1.124	.281	.18	.9450
Error	32	48.860	1.527		

ANOVA Table for Baseline Shift in Force

Source of Variation	df	Sum of Squares	Mean Square	F	p <
Subjects	8	56.803	7.100		
Feedback Mode (M)	1	13.015	13.015	.994	.3479
Error	8	104.735	13.092		
Force Magnitude (F)	**2**	**33.900**	**16.950**	**7.599**	**.0048**
Error	16	35.688	2.230		
MF	**2**	**30.426**	**15.213**	**8.35**	**.0033**
Error	16	29.156	1.822		
Ramp Speed (R)	2	1.208	.604	.345	.7135
Error	16	28.039	1.752		
MR	2	.081	.041	.054	.9478
Error	16	12.108	.757		
FR	4	3.327	.832	1.15	.3527
Error	32	23.220	.726		
MFR	4	4.461	1.115	1.73	.1662
Error	32	20.544	.642		

N91-20656

Autonomous Proximity Operations Using Machine Vision For Trajectory Control and Pose Estimation

Dr. Timothy F. Cleghorn
Mail Code PT4
NASA/Johnson Space Center
Houston, TX. 77058

Dr. Stanley R. Sternberg
Machine Vision International, Inc.
912 N. Main Street
Ann Arbor, Michigan, 48104

Abstract

A machine vision algorithm has been developed which permits guidance control to be maintained during autonomous proximity operations. At present this algorithm exists as a simulation, running upon an 80386 based personal computer, using a ModelMATE CAD package to render the target vehicle. However, the algorithm is sufficiently simple, so that following off-line training on a known target vehicle, it should run in real time with existing vision hardware. The basis of the algorithm is a sequence of single camera images of the target vehicle, upon which radial transforms have been performed. Selected points of the resulting radial signatures are fed through a decision tree, to determine whether the signature matches that of the known reference signature for a particular view of the target. Based upon recognized scenes, the position of the maneuvering vehicle with respect to the target vehicle can be calculated, and adjustments made in the former's trajectory. In addition, the pose and spin rates of the target satellite can be estimated using this method.

INTRODUCTION

In order to perform a rendezvous and docking operation in space, it is necessary to determine the attitude and attitude rates of the target vehicle, as well as the relative position and trajectory of the maneuvering craft with respect to that target vehicle. These parameters are obtained currently by using Shuttle astronauts' eyes to guide the maneuvering craft to the desired position so that a grapple with the Shuttle Remote Manipulator System,(RMS), can be performed by a crew member. In the future, it will be desirable to perform these operations with increasing degrees of autonomy; particularly satellite servicing, and Lunar and Martian orbiter rendezvous. In order to do this, a full array of sensors will be required; however it is likely that vision will remain as the major source of input data. One of the chief drawbacks of any sensing system based upon vision data is the sheer number of those data, with the correspondingly long computation times required to process the input. It is therefore very important to develop methods of data compression which permit analyses in keeping with the time scale defined by the characteristic motions of the target/sensor system in question. An algorithm has been developed which permits small errors or drifts in trajectory to be identified and corrected, based upon the view of the target vehicle as seen by a single camera on a maneuvering craft. This algorithm is demonstrated on a PC computer with EGA or VGA graphics.A CAD/CAM system, (ModelMATE, by Generic Software, Inc.), has been used to model the target vehicle. Current vision hardware includes Imaging Technology's PC-Vision frame grabber mounted in a COMPAQ 286, and a Sony XC-57 CCD camera. This is scheduled to be upgraded to an ASPEX PIPE machine attached to a Sun 4in the near future. High fidelity graphics models will be included, and solid models will also be employed. Figure 1 illustrates one view of the target, a (somewhat fanciful) Hubble Space Telescope. It is assumed that the target object is located within the field of view of the camera, and that the target is recognized by the system; i.e., target identification is not the issue, although the techniques described herein could well be used for that purpose also. This algorithm utilizes the radial signatures of a sequence of images to determine a calculated position and trajectory for the maneuvering craft.

The complete program consists of two parts: an off-line training phase, and a series of run-time calculations, as the maneuvering craft approaches the target vehicle. The training phase presupposes the existence of an accurate three-dimensional CAD model of the target vehicle, and typically runs for two days on an 80386 type computer for the level of accuracy used in this work. The training phase consists of the building of decision trees which permit the association of a radial signature of the target's image with an angular orientation of the target vehicle with respect to the maneuvering craft. Details of the training process will be presented in the next section.

Following the off-line training, a "desired" rendezvous trajectory is selected. It is assumed that the angular orientation of the target craft is known to within an accuracy of about 20 degrees at some initial time t0. An angular normalization is made around the camera-target axis to align the image axes with those used during the training phase. Radial signatures of successive images are extracted as the maneuvering vehicle attempts to fly its desired trajectory, and these signatures are normalized to correspond to those used during the training phase. Points on these radial signatures are fed into a decision tree to determine whether the camera "recognizes" the view. It is normal that for each image,several adjacent views are recognized. Based upon the linear extent of an image compared to a reference image, the apparent distance between the camera and the target can also be calculated. Thus a sequence of images generates a "point cloud", through which a curve or apparent trajectory can be fit. This permits the next segment's trajectory to be predicted, and corrections to be made to drive it closer to that which was planned originally. In addition, or as an alternative, is possible to calculate the target vehicle's attitude and attitude rates. These are necessary parameters for an autonomous docking to be performed.

PROCEDURE

Reference Frame Construction

During both the training and production phases of the algorithm, the relative positions of the target and observing crafts are defined by constructing a geodesic sphere around the target. This virtual sphere is attached to the target vehicle, and the observing craft moves on or outside of the surface. If the observing craft moves inside of the geodesic, a new sphere must be constructed in order to account for distortion. It will be assumed that the geodesic encloses the entire target vehicle. For each node, or line intersection on the sphere's surface, a characteristic view is stored. Actually, using a relatively new technique which will be discussed below, the critical information for a given node is compressed to be only a few numbers, typically six to eight. These numbers are stored in a hierarcical decision tree for each node on the sphere. The geodesic sphere is constructed by repeated bisections of a regular icosahedron, (a twenty-sided polydedron). Each surface of the icosahedron is an equilateral triangle. By connecting midpoints of the edges of each triangle, four new triangles are constructed. If the icosahedron is considered to be the zeroth order sphere, the number of surfaces on an ith order sphere is given by:

1) $\quad nfaces_i = nfaces_0 * 4^i$

where $nfaces_0 = 20$

In terms of the i-1 order geodesic,

1a) $\quad nfaces_i = 4 * nfaces_{i-1}$

Similarly, the number of edges of an ith order geodesic is given by:

2) $\quad nedges_i = 1.5 * nfaces_i$

Each triangle on the surface of the geodesic has three edges, each one of which is shared by one adjacent triangle, hence the factor 1.5. The number of vertices, or nodes is given by:

3) $\quad nnodes_i = nnodes_{i-1} + nedges_{i-1}$

where $nnodes_0 = 12$ for the zeroth order icosahedron.

The density of nodes will determine both the accuracy of the pose calculation and the computer time required for training. It was found that a third order geodesic, with 642 nodes and 1280 faces was a good compromise between accuracy and computing time.

Signature Construction

Having established a coordinate frame, it is necessary to find those parameters which will identify a view of the target uniquely from any location within the space on or outside of the surface of the geodesic. Binary thresholding permits the most rapid computation. In addition to providing the radial signature of the target vehicle, as described below, the binary image allows calculation of the distance of the camera from the target. During training, the areal extent, Aref, of the target image is recorded for each of the 642 nodes. The linear distance, from the centroid of the target image to the camera is given by:

4) $\quad d_{calc} = d_{ref} * \text{sqrt}(A_{ref}/A_{obs})$

where d_{ref} is a reference distance, (the radius of the geodesic), and A_{obs} is the observed area of the target image.

Both the training and the on-line or production portions of the program utilize the radial transform to reduce the raw data from the image of the target vehicle to a level which can be dealt with by an AT-class machine. The implementation of the radial transform is a fairly straight-forward procedure, which has been coded in C in order to conform to several available hardware machine vision systems. The transform itself consists first of locating the centroid of the binary image of the target vehicle. Care must be taken to insure that the binary image outline corresponds to the grey level outline of the vehicle, and in fact one future project will be the development of software to permit the binary image to be reconstructed should this correspondence fail due to lighting or other problems. Following location of the centroid, the radial distances to the outermost edge of the binary image is measured. The simulation demonstration uses 294 radial measurements, corresponding to the 294 vertical bins on an EGA graphics screen. The hardware implementation for the PC-Vision board uses 360 radial bins, starting at East,(bin 0), and running counterclockwise. The radial signature of the target is obtained by plotting these distances as a function of bin number. (Figures 2a-b).

Decision Tree Construction

The 294 or 360 bins still represent too large a number of data to analyze,either during the training or on-line phases of the program. For each of the 642 nodes of the geodesic we wish to have no more than about 10 characteristic features which will identify the node. It is assumed that the relative angular position is known approximately, so that there is no ambiguity between polar symmetric nodes. Additionally, a statistical approach is taken: it is desired that each node's state be classified correctly between 95% and 98% of the time.

For the training phase, each of the 642 nodes is labeled. The camera is assumed to be located on the surface of the geodetic. In order to train for a specific node, the radial signature of that node, plus those for a number of surrounding points are obtained. The surrounding points are selected to be the mid-points of the edges, up to three edge lengths away from the central node, (Figure 3). It is desired that all views up to and including one edge length's distance from the central node be recognized, and all views between one and three edge lengths not be recognized. As can be seen in Figure 3, 73 radial signatures are extracted for each node, of which 19 are "in", and 54 are "out". This operation can be thought of as applying a series of perturbations to the target object. The views seen by the camera will "wobble" about a central axis.

It is desired to select those particular radial bins which will identify the view from the given node most rapidly. A decision tree must be constructed, the terminal branches of which label the node as being "in" or "out", at some level of certainty. There are two general types of classifiers which can be used to separate a data set into components. These include single stage classifiers, such as Bayes linear and quadratic classifiers, Fisher's linear classifier, thresholding the principal feature, or thresholding a component. All of these classify the data into two or more classes in a single step. The present work uses a new technique of classification called a hierarchic classifier. This method can be described as a binary decision tree, in which each terminal branch represents one pattern class, and the non-terminal nodes of the tree represent a collection of classes. The root node represents the entire collection of classes. When an unknown datum enters the hierarchic classifier at the root node, a decision rule associated with the root node is applied to it to determine the next node to which it should go. This process is repeated until a terminal node is reached. Each terminal node has an associated class to which that datum is assigned.

In order to implement a hierarchic classifier a decision rule must be constructed for each node of the tree. A decision rule is a single-stage classifier, such as any one of the types mentioned above. The simplest of these is that which thresholds a component of the data. Thus the construction of the entire decision tree involves three steps: choosing the decision rules at each node of the tree, finding different ways of branching from a non-terminal node to its child nodes, and finding the termination condition for the branching process. The branching condition at each non-terminal node is based on a criteria of minimum entropy or minimum classification error. At each node of the tree, consider a threshold for each data component for all samples of the data. This threshold partitions the data into two classes, those with component values less than the threshold, and those with values greater. The entropy is then computed for left and right partition classes. If the decision rule is effective, these values will be significantly different. If L_i is the number of feature vectors in category i classified to the left child, and R_i is the number classified to the right, the entropy H_i is defined as:

$$5)\quad H_i = = L_i * \ln(L_i/L) + R_i * \ln(R_i/R) - (L_i + R_i) * \ln((L_i + R_i)/(L + R))$$

where $L = \quad L_i$, and $R = \quad R_i$. The index i takes on the values "on" and "off".

The entropy is computed for all components of the data and for all thresholds that can partition the data into two classes at each node of the decision tree. The threshold and the component which gives the minimum entropy are considered to be the appropriate ones for that node.

The branching process is terminated when one of the following conditions is met. If the number of samples falls below a certain minimum, the entropy calculation is meaningless. If all samples at a particular node fall in one category, the branching process is stopped, and the class of the node is assigned to that category. Also, if the entropy calculated by equation (1) falls below a certain minimum, there is no significant difference between right and left partitions. In this case, the right and left children are merged into one node. It was found that by using these criteria to determine when to terminate the branching process, the view recognition accuracy was consistently within the desired 95 and 98 percent rate.

The decision tree can be represented in the computer as a series of if-then-else statements. Consider a set of data with three components, (r1, r2, r3). Five samples have component values as follows:

Sample	r1	r2	r3	Category
s1	0.6	1.0	1.0	2
s2	0.4	1.0	0.8	1
s3	0.6	1.0	0.8	2
s4	0.6	1.2	0.8	1
s5	0.6	4.4	0.8	2

Table I

The categories are assigned here simply as left child or right child at the terminal node. Figure 4 illustrates the resulting decision tree. The thresholds are given for each non-terminal node, and the resulting classification appears at the terminal node for each sample. The advantages of the decision tree approach are first that it identifies which components are important, and second, it is faster than the single-stage classifier techniques once the training phase has been completed. It also can be expressed readily in an Expert System format:

```
IF (r2 < = 1.1) {
  IF (r3 < = 0.9) {
    IF (r1 < = 0.5)
      ASSIGN Category = 1 ; terminal node left
    ELSE
      ASSIGN Category = 2 ; terminal node right
    }
  ELSE
    ASSIGN Category = 2
  }
ELSE
  {
    IF (r2 < = 2.3)
      ASSIGN Category = 1
    ELSE
      ASSIGN Category = 2
  }
```

Figure 5 illustrates two decision trees constructed for separate nodes on the geodesic for the Hubble Space Telescope. "T" stands for a terminal node. If the view is recognized, the value assigned to the terminal node is 1; otherwise it is 0. The radial vector is the first number in the inequality, the threshold value is the second. Thus "2 #156 < = 455" can be read as "If radial vector #156 has a value less than or equal to 455, then...." The initial integer "2" refers to the level within the decision tree. There are two points to observe in Figure 5. First, once a terminal node has been encountered, the calculation is finished. This speeds up the algorithm considerably. Second, note that one of the decision trees is quite long compared to the other. To understand physically what is occurring, consider a thin flat plate of somewhat irregular shape. If viewed nearly edge on, a slight wobble or perturbation will cause the outline or signature of the plate to change significantly. However, if viewed from a point nearly perpendicular to the plate, the same amount of wobble will change the outline or signature only slightly. Thus some viewing directions are vastly simpler than others to identify. The price paid to use the hierarchic classifier is that a decision tree must be constructed for each of the 642 nodes on the geodesic surface. The total time needed to do this was about two days, using an AT-class machine.

Decision Tree Application

In the preceding section, the procedures used to train the classifier have been discussed. Following the training, the second phase of the algorithm takes place, namely its application using images from unknown directions. It must be assumed however, that the target vehicle's pose is known to about 20 degrees at the initial time t_0; otherwise the time it takes to locate a group of recognized "on" nodes will exceed that which it generally takes for the pose to change to some new, and still undetermined value.

There are two initial corrections which must be applied to each of the images. The first of these, the distance correction, has already been discussed, (equation 4). In some cases it was necessary to add a correction for the difference in focal length between the reference and the flight images. The distance equation then becomes:

4a)

$$d_{calc} = d_{ref} * \mathrm{sqrt}(A_{ref}/A_{obs}) * (\mathrm{image\_focal\_length} / \mathrm{reference\_focal\_length})$$

The other initial correction is for rotation about the line-of-sight between the crafts. Again, this assumes an approximately known initial pose.

Having made these corrections, the radial signature of the unknown image is is extracted, and applied to the decision trees of all of the nodes in the neighborhood of the approximate position on the geodesic. If in fact the camera lies somewhere within this region, some of the nodes should recognize the view, that is, they should be "turned on". One of the major advantages of the hierarchic classifier approach is that with several of the nodes being activated simultaneously, if one or two should be missed, the position can still be calculated. Thus an element of robustness against bad lighting conditions, reflections and background is built into the method. Using the distance correction obtained from equation (4), a calculated position in three dimensional space is found for each "on" node. For each image, there are typically five such points. As the maneuvering craft moves with respect to the target, the process is repeated, with new images generating new points, forming what is referred to as a point cloud along the trajectory of the maneuvering craft. The position of the maneuvering vehicle is then calculated using a multi-dimensional minimization procedure called the "Downhill Simplex" algorithm. For a discussion of this method see Press, et al, 1988. This can be thought of as analogous to a four dimensional best fit through the point cloud. The orbits of the maneuvering vehicle were calculated in segments, in order to be able to determine how far that craft was from the desired path. For the cases of circular or spiral rendezvous, one radian segments were chosen. This permitted drift errors to be detected, and the path to be adjusted before the errors became too great. Thus new paths were planned for successive segments, allowing the maneuvering craft to stay close to the desired trajectory.

In addition to the circular and spirial trajectories, this was done using an actual Space Shuttle V-Bar approach trajectory. This required using equation (4a) to determine the distance correction, and image distortion also became a serious problem. As for the circular and spiral cases, it was possible to correct for distortion to some extent by constructing a virtual geodesic with a smaller radius, even to the point of enclosing just a portion of the target vehicle. This relearning obviously becomes very expensive computationally, and really defines one of the limits of usefulness of the algorithm.

In addition to being able to calculate the trajectory for the maneuvering craft, it is possible to calculate the attitude or pose, and attitude rates for the target vehicle. In fact, if the two vehicles are at constant distance from each other in some global coordinate system, the attitude/attitude rate calculation is entirely equivalent to the maneuvering vehicle trajectory determination. The six numbers describing the pose and spin of the target vehicle are needed for an autonomous docking or grappling to occur. Therefore, the hierarchic classifier approach has a much wider potential application than was originally intended.

CONCLUSIONS

A new method of determining the trajectory of a maneuvering craft with respect to a target vehicle has been described. This method utilizes a hierarchic classifier with input data from a single camera, to calculate either the trajectory of the maneuvering craft, or to determine the pose and spin parameters of the target vehicle, or both. The advantages of this method are that it is faster during on-line calculations than the single-stage classifier methods, it is robust with respect to partial or noisy input data, and it identifies the important components of the target image. The algorithm also runs on commonly available computer systems.

Currently, the algorithm exists as a simulation demonstration, with some pieces having been ported to a hardware machine system. It is planned to continue this porting process, and demonstrating the algorithm using physical models, as well as actual images of satellites in space. This latter will permit testing of the robustness of the algorithm; both Earth and space backgrounds will appear in the images, as well as shadows and reflections on the target vehicle.

ACKNOWLEDGEMENTS

This work is the direct result of SBIR Contract NAS 9-17814 with Machine Vision International, of Ann Arbor, and subcontract to the University of Washington, in Seattle. Without the full and generous cooperation of personnel from both organizations, this work would not have been possible. Special thanks go to Mr. William Dargel, of MVI, and to Professors Robert Haralick and Linda Shapiro, of the University of Washington.

REFERENCES

Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, "Numerical Recipes in C", Chapter 10, p305ff Cambridge University Press, 1988

Sternberg, Stanley R., "Integrated Computer Vision Research for SpaceConstruction -- FINAL REPORT Vol I", Machine Vision International, Inc., Ann Arbor, Michigan, November, 1989

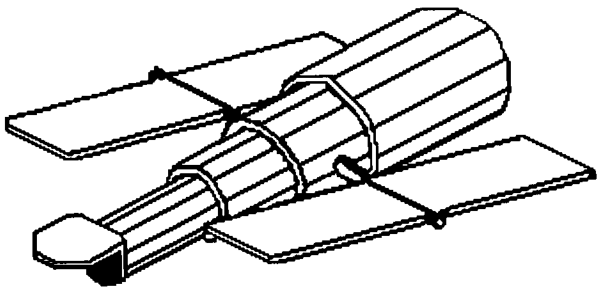


FIGURE 1

Space Telescope

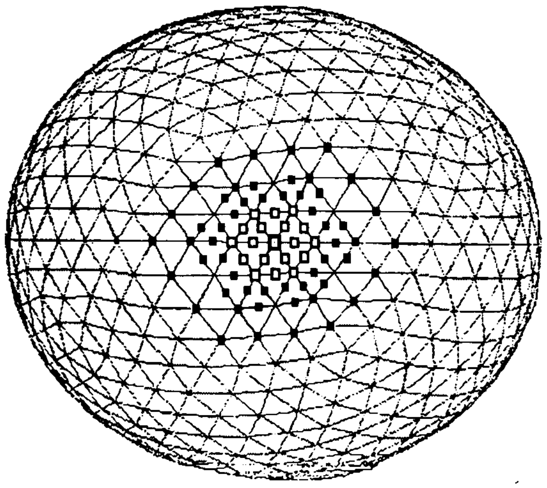


FIGURE 3

Training view positions for a node. There are 19 "ON" positions (open boxes), and 54 "OFF" positions (closed boxes) for training each of the 642 nodes on the geodesic.

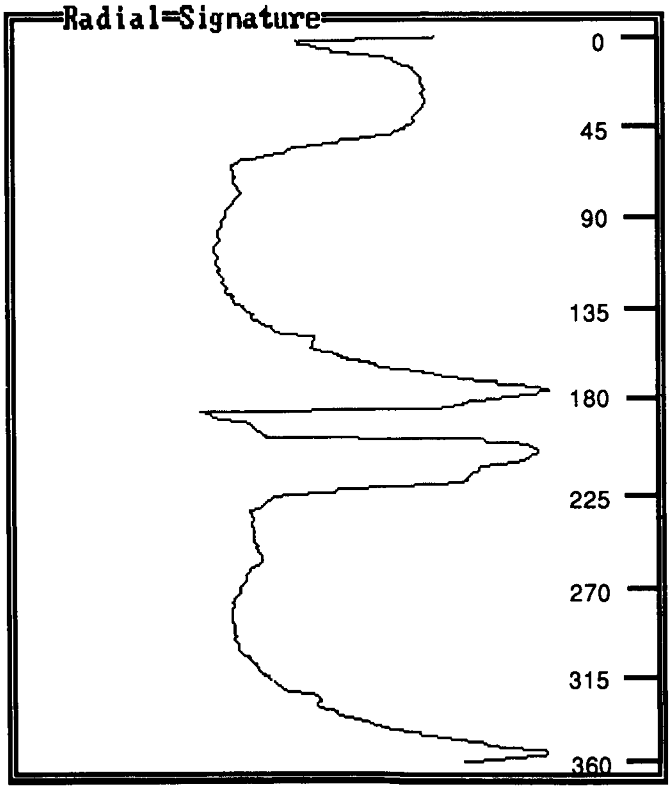


Figure 2a

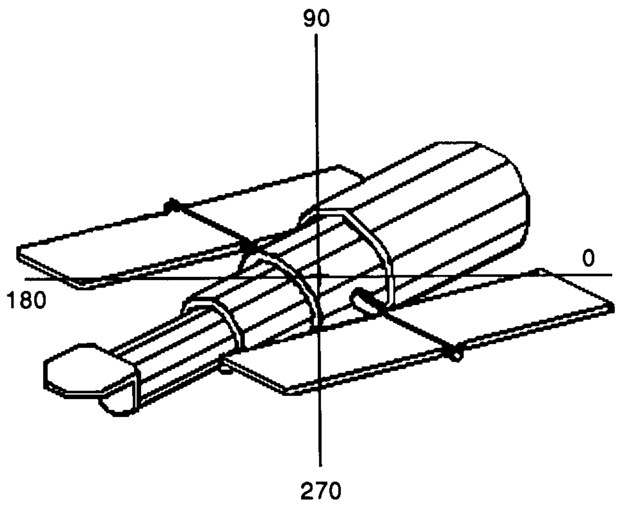


Figure 2b

The radial signature, (Fig 2a), is obtained from the binary image of Figure 2b, by measuring the radial distance from the centroid (+) to the outermost edge of the object.

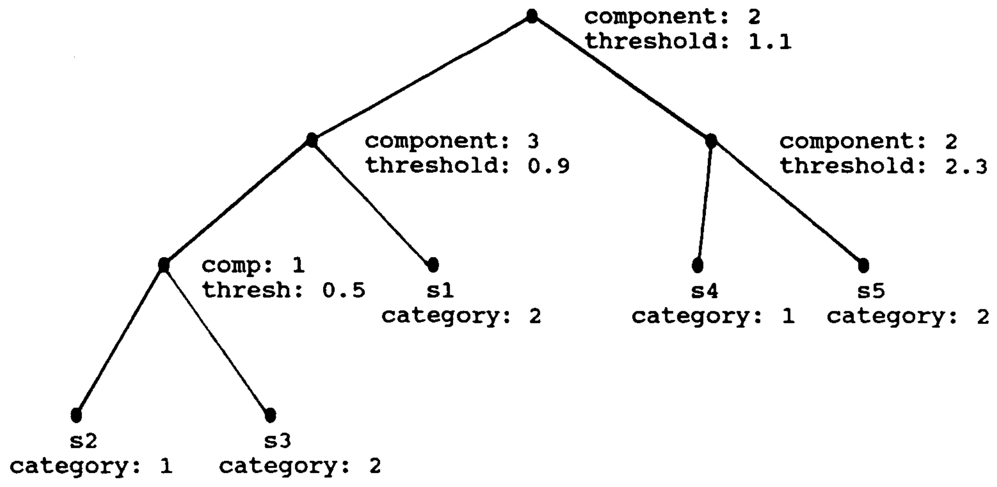


Figure 4

A decision tree for the data in Table I is illustrated. Left branches represent component values less than the threshold at a branching node, whereas right branches represent component values above the threshold. The sample is assigned to the category (left or right) at the terminal node.

```
Node   12

0 #187 <= 486
  1 #259 <= 459
    2 #7 <= 1293
      3 T 1
      3 T 0
    2 #18 <= 983
      3 #69 <= 380
        4 T 1
        4 T 0
      3 T 1
  1 #189 <= 349
    2 T 1
    2 T 0
```

```
Node 13

0 #149 <= 830
  1 T 0
  1 #176 <= 616
    2 #150 <= 1146
      3 #170 <= 515
        4 T 0
        4 #293 <= 1044
          5 #31 <= 480
            6 T 0
            6 #117 <= 415
              7 #182 <= 477
                8 T 0
                8 T 1
              7 T 1
          5 T 0
      3 T 0
    2 #286 <= 1444
      3 T 0
      3 #176 <= 660
        4 #0 <= 995
          5 T 1
          5 T 0
```

Figure 5

Two decision trees used in the operational phase. One is short, representing a relatively unambiguous view of the target, whereas the other is long, which indicates that the view from that node is difficult to recognize. The node numbers do not indicate relative locations of the two views.

N91-20657

BERTHING SIMULATOR FOR SPACE STATION AND ORBITER

Dr. Sam Veerasamy, P. E.
Principal Engineer
Simulation Development Department
Lockheed Engineering and Sciences Company
2400 Nasa Road 1, C83, Houston, TX 77058

ABSTRACT

The development of a real-time man-in-the-loop berthing simulator is in progress at NASA Lyndon B. Johnson Space Center (JSC) to conduct a parametric study and to measure forces during contact conditions of the actual docking mechanisms for the Space Station Freedom and the orbiter. In berthing, the docking ports of the space station and the orbiter are brought together using the orbiter robotic arm to control the relative motion of the vehicles. The berthing simulator consists of a dynamic docking test system (DDTS), computer system, simulator software, and workstations. In the DDTS, the space station and the orbiter docking mechanisms are mounted on a six-degree-of-freedom (6 DOF) table and a fixed platform above the table. Six load cells are used on the fixed platform to measure forces during contact conditions of the docking mechanisms. Two Encore Concept 32/9780 computers are used to simulate the orbiter robotic arm and to operate the berthing simulator. A systematic procedure for a real-time dynamic initialization is being developed to synchronize the space station docking port trajectory with the 6 DOF table movement. The berthing test can be conducted manually or automatically and can be extended for any two orbiting vehicles using a simulated robotic arm. The real-time operation of the berthing simulator is briefly described in this paper.

INTRODUCTION

Berthing is the joining of docking ports of any two orbiting vehicles using a robotic arm to control the relative motion of the vehicles while docking uses on-board propulsion system to perform the same task. During and after construction of the space station, it will be necessary to transport large quantities of materials, consumables, crew, life supporting items, etc., to and from the space station. Berthing is preferred over docking, because relative velocities and impact loads are smaller than those for docking. Further, there will be no plume impingement on the space station during berthing. Since the space station is to be manned, on-board manual control that uses direct control and/or remote visual cues appears to be the simplest method of control for the berthing maneuver.

The real-world berthing operation is shown in figure 1. The space station is captured by the end effector of the orbiter robotic arm or the 6 DOF remote manipulator system (RMS). The orbiter and the space station docking ports are denoted by DP1 and DP2 respectively. The berthing operation starts after the RMS captures the space station. The DP2 is the point of resolution (POR) for the orbiter RMS. The RMS is commanded to control the relative distance and attitude of the docking ports. The docking ports slowly come to initial contact for soft latch, and finally hard latch. During the initial construction phase of the space station, the orbiter RMS will be used for berthing operation. After construction is completed, a 7 DOF space station RMS will be used for berthing.

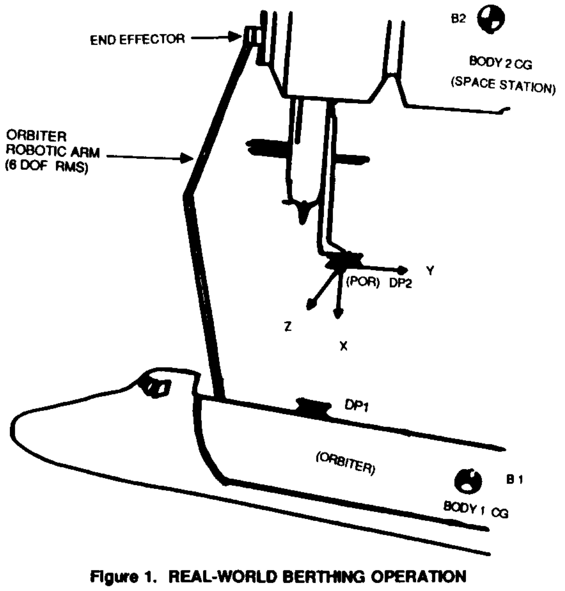


Figure 1. REAL-WORLD BERTHING OPERATION

A real-time berthing simulator is being developed at NASA/JSC to conduct a detailed parametric study of the actual docking mechanisms for the space station and the orbiter. The berthing study will examine position and attitude misalignments, linear and angular velocities, contact forces and latching of the docking mechanisms. The simulator consists of a DDTS, two Encore Concept 32/9780 computers, simulator software, and workstations. A closed circuit television (CCTV) monitoring system provides views of the docking mechanisms and alignment aids. The simulator software models consist of the RMS control system, the RMS dynamics, the RMS geometry, the docking port relative geometry, the actuator command, the load cell forces and moments, and the physical characteristics of the orbiter, the 6 DOF table and a particular space station configuration. The berthing simulation models are designed to grow as new requirements develop. The hardware and software components and the real-time berthing operations are briefly described in this paper.

DDTS

Figure 2 shows the DDTS used for berthing operation and a perspective view of the DDTS is shown in figure 3. The 6 DOF table is driven by six coupled hydraulic actuators(1). The actual space station and orbiter docking mechanisms to be tested are mounted on the 6 DOF table and the fixed platform above the table. The position and velocity of the table are controlled by commanding the six hydraulic actuators. The contact forces are measured by three pairs of calibrated load cells, mounted between the body 1 docking ring and body 1 mounting ring as shown in figure 4. The load cells are mounted in equidistance around the ring to obtain a three dimensional force vector by resolving the individual load cell forces. The load cell output voltage is assumed to be proportional to the applied force and is zeroed to compensate the orbiter docking mechanism hanging from the body 1 fixture. The output of the load cells are provided to the load cell forces and moments model through analog to digital converters. The real-time software model is executed at 50 Hz.

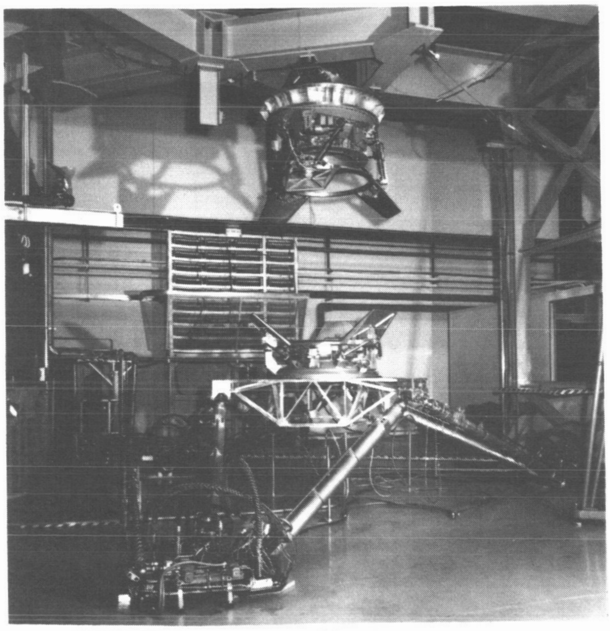


Figure 2. DYNAMIC DOCKING TEST SYSTEM

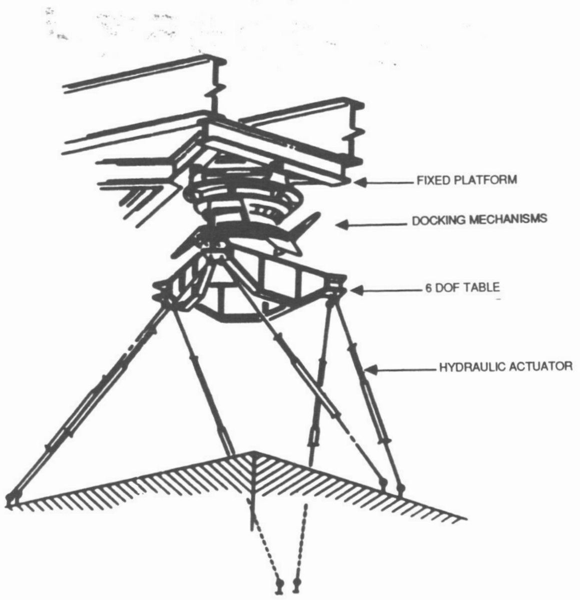


Figure 3. PERSPECTIVE VIEW OF THE DDTS

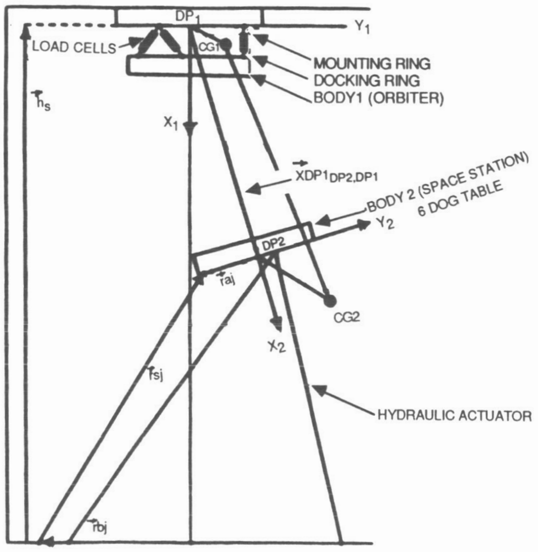


Figure 4. DDTS GEOMETRY

The DDTS can be safely operated up to 3,000 lb impact force with 2 lb resolution. The height of the 6 DOF table can be controlled vertically from 4.24 ft to 14.63 ft with a resolution of 0.006 in by commanding the hydraulic actuators. The attitude (roll, yaw and pitch) of the 6 DOF table can be controlled within ±20°. When the docking mechanisms are not installed, the table can travel a vertical distance of 10 ft. The sidewise movement of the table can be controlled within ± 3 ft.

RMS SOFTWARE

The RMS is modeled as a six joint flexible robotic arm. The RMS dynamics and control system models provide a reasonable accuracy at docking port contact conditions and maintain stability for the simulation. The software is already in use in a real-time simulation(2). The RMS model is tailored to meet the requirements for the berthing simulation (3).

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

In a real-time operation, RMS dynamics and control system models run at 50 Hz and 25 Hz respectively. Malfunctions and capture/release transitions are not simulated. The effect of non-linear torque is assumed to be negligible. As gravity forces and gravity gradient forces are small compared to contact forces and moments, the gravity gradient effects are neglected. The docking port relative geometry is computed using the orbiter RMS POR position and attitude.

WORKSTATIONS

The simulator workstation consists of an 80386-based 33 MHz computer with a math co-processor, and printers to monitor and dump real-time data and post process data. This workstation supports dry runs, man-in-the-loop runs, berthing programmed test input runs, simulator readiness tests and health status tests. During dry runs, the real-time software system is operational, but the hardware system (hydraulic actuators, load cells etc.) is not powered.

The RMS workstation consists of two closed circuit television (CCTV) monitors, two hand controllers and a graphics system. The graphics system displays the difference between the simulated and the actual RMS commands as well as other status information from the simulation. This workstation provides the capability for a pilot or a RMS operator to perform a simulated berthing maneuver and monitor its progress.

DOCKING PORT RELATIVE GEOMETRY MODEL

The docking port relative geometry model receives the orbiter RMS and the space station data from the RMS geometry model and the physical characteristics of the orbiter, the space station, and the 6 DOF table. This model computes the relative position and attitude of the two docking ports for berthing operation. The real-time model is executed at 50 Hz. The output of the docking port relative geometry is sent to the actuator command model.

Figure 5 shows the docking port relative geometry. The RMS dynamics model provides POR distance ($\vec{X}BA_{DP2,BA}$) in orbiter body axis coordinates. The orbiter docking port distance ($\vec{X}SR1_{DP1,SR1}$) is calculated in orbiter structural coordinates based on the orbiter physical characteristics. The relative distance between the docking ports in DP1 coordinates ($\vec{X}DP1_{DP2,DP1}$) is calculated as

$$\vec{X}DP1_{DP2,DP1} = [DP1, B1]\,[B1, BA]\,\{\vec{X}BA_{DP2,BA} - [BA,SR1]\,\vec{X}SR1_{DP1,SR1}\}$$

where

[DP1, B1]	is the transformation matrix from B1 (body1 CG) to DP1 coordinates,
[B1, BA]	is the transformation matrix from BA to B1 coordinates,
$\vec{X}BA_{DP2,BA}$	is the POR distance in orbiter body axis coordinates provided by the RMS dynamics model,
[BA, SR1]	is the transformation matrix from SR1 to BA coordinates,
$\vec{X}SR1_{DP1,SR1}$	is the orbiter docking port distance in SR1 coordinates.

The RMS dynamics model computes POR attitude in RMS arm reference coordinates, A. Therefore, the relative attitude between the docking ports, [DP1, DP2], is calculated as

$$[DP1, DP2] = [DP1, B1]\ [B1, A]\ [A, DP2]$$

where

[B1, A]	is the transformation matrix from A to B1 coordinates,
[A, DP2]	is the transformation matrix from DP2 to A coordinates.

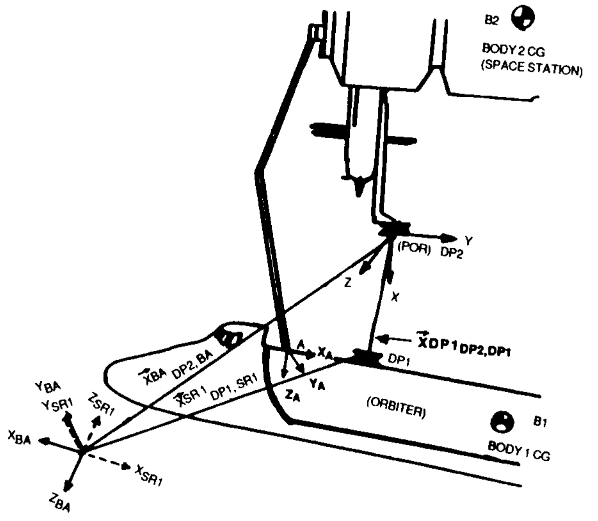


Figure 5. DOCKING PORT RELATIVE GEOMETRY

ACTUATOR COMMAND MODEL

The actuator model calculates command signals for the six hydraulic actuators. Each actuator length ($\vec{r}_{sj}$) is calculated as a function of the DDTS geometry, shown in figure 4, and the relative docking port geometry model.

$$\text{i.e.,}\quad \vec{r}_{sj} = -\vec{r}_{bj} + \vec{h}_s + \vec{X}DP1_{DP2,DP1} + [DP1,DP2]\,\vec{r}_{aj}$$

where

j	represents actuator number (j =1,2,...6),
$\vec{r}_b$	is the distance between the intersection of X1 with the base and the actuator floor pivot point in body 1 coordinates system,
$\vec{h}_s$	is the height between the hydraulic actuator floor pivot point and DP1 in the body 1 coordinates system,

$\vec{r}_a$ is the distance between DP2 and the actuator table pivot point in the body 2 coordinates system.

The actuator stroke or the change in actuator length (l_{sj}) from its initial (l_{oj}) position is calculated as

$$l_{sj} = (r^2_{sjx} + r^2_{sjy} + r^2_{sjz})^{1/2} - l_{oj}$$

where r_{sjx}, r_{sjy}, r_{sjz} are x,y,z components of $\vec{r}_{sj}$.

The position and velocity of the 6 DOF table are controlled by commanding the actuator stroke. The real-time actuator model is executed at 50 Hz.

DYNAMIC INITIALIZATION

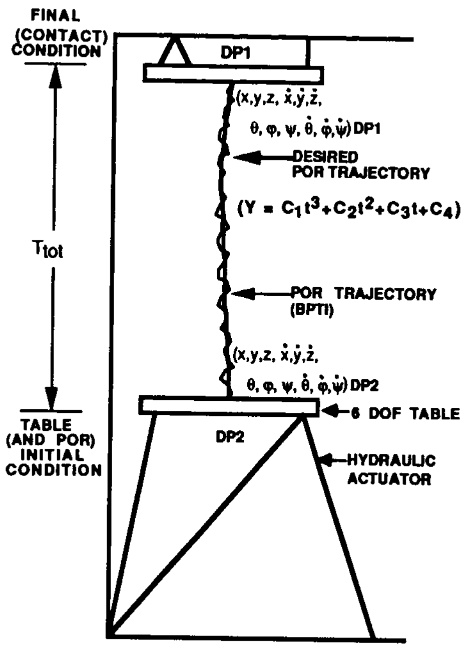
There are two major existing hardware and software limitations for the berthing simulation. First, the maximum table travel distance in the DDTS may not be sufficient for the simulated POR travel to achieve the desired impact velocity. Second, the simulated POR in the RMS model must start with zero velocity (i.e., the RMS arm travel may require about 50 ft to attain the desired impact velocity, but the 6 DOF table will not have the capability of achieving 50 ft travel due to physical limitations of the DDTS hardware set-up). A testing capability is being developed to overcome the limitations by using dynamic initialization of the 6 DOF table and the simulated POR.

Initially, the 6 DOF table and the simulated POR will be tried to move together from the table initial condition (IC) to the final contact condition. The RMS POR initial condition is synchronized in such a way that the POR starts from the table initial condition as shown in figure 6(a). This method is called the dynamic initialization with no offset condition. The initial condition $(x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi})$ DP2 and the final condition $(x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi})$ DP1 are known (i.e., x, y, z are positions in Cartesian coordinates, $\dot{x}$, $\dot{y}$, $\dot{z}$ are velocities, θ, ϕ, ψ are Euler angles, and $\dot{\theta}$, $\dot{\phi}$, $\dot{\psi}$ are Euler angular rates). A total travel time (T_{tot}) for the table is assumed. The trajectory of each parameter (x) is calculated based on the four boundary conditions (x_{DP1}, $\dot{x}_{DP1}$, x_{DP2}, $\dot{x}_{DP2}$) and the travel time (t) varies from zero to T_{tot}. A polynomial of third degree is assumed for the table (and the POR) trajectory.

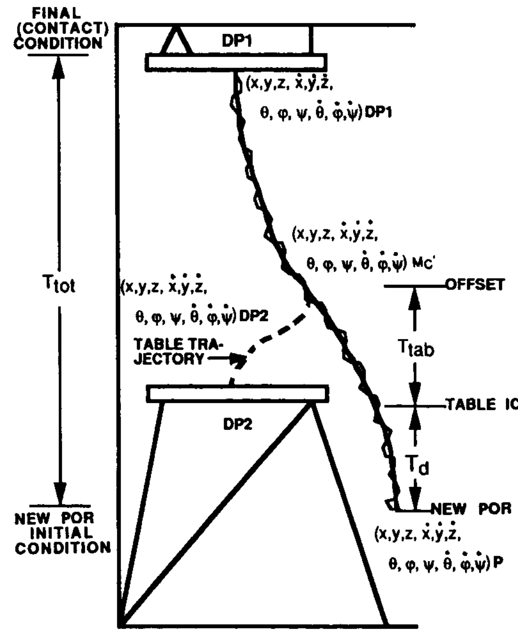
i.e., $Y = f(t) = C_1 t^3 + C_2 t^2 + C_3 t + C_4$

where Y is the trajectory of x, y, z, θ, ϕ, ψ, C_1, C_2, C_3, C_4 are constants.

First, the table travel is checked to ensure that the table velocity and Euler angles will be less than the safe limit set for the simulator abort conditions. Second, the RMS end effector velocity, calculated from the table velocity, is checked to be less than the maximum end effector velocity limit. In a dry run, the POR and the table trajectories will be checked for a successful real-time operation without triggering any singularity for the arm and other simulator abort conditions. During the dry run, the contact forces simulated by a second order spring model, the RMS health monitor functions, reach limit, control singularity etc., will be monitored. If the dry run is not successful, the total travel time may be increased to reduce the table velocity and the dry run is repeated.


FINAL (CONTACT) CONDITION
DP1
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) DP1
DESIRED POR TRAJECTORY
(Y = C1t3+C2t2+C3t+C4)
Ttot
POR TRAJECTORY (BPTI)
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) DP2
6 DOF TABLE
TABLE (AND POR) INITIAL CONDITION
DP2
HYDRAULIC ACTUATOR

(a) NO OFFSET


FINAL (CONTACT) CONDITION
DP1
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) DP1
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) MC
OFFSET
Ttot
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) DP2
TABLE TRAJECTORY
Ttab
TABLE IC
DP2
Td
NEW POR IC
NEW POR INITIAL CONDITION
(x,y,z, ẋ,ẏ,ż, θ, φ, ψ, θ̇, φ̇, ψ̇) P

(b) OFFSET

Figure 6. DYNAMIC INITIALIZATION

If the POR travel distance is not sufficient to achieve the desired impact velocity, a new POR initial condition (IC) at a point P will be selected, as shown in figure 6(b), to increase the POR travel distance only. An offset point, $(x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi})$ M_C', will be selected on this new POR trajectory. The table trajectory is then calculated based on the boundary conditions of the table (DP2), offset condition (M_C'), and assumed table travel time (T_{tab}). A polynomial of third degree is used similar to the calculation of the POR trajectory. Once again, the new POR and the table trajectories will be checked for a safe operation without triggering simulator abort conditions. The POR will be commanded to move well in advance and then the table movement will be started with a time delay (T_d) in such a way that the table and the POR will meet at the offset condition (M_C') with the same velocity and Euler angular rate. This method of synchronization is called the dynamic initialization with offset condition. Both the POR and the table continue to move together until the initial contact occurs with DP1. If the dry run results are within tolerance, a complete software and hardware real-time test will be performed.

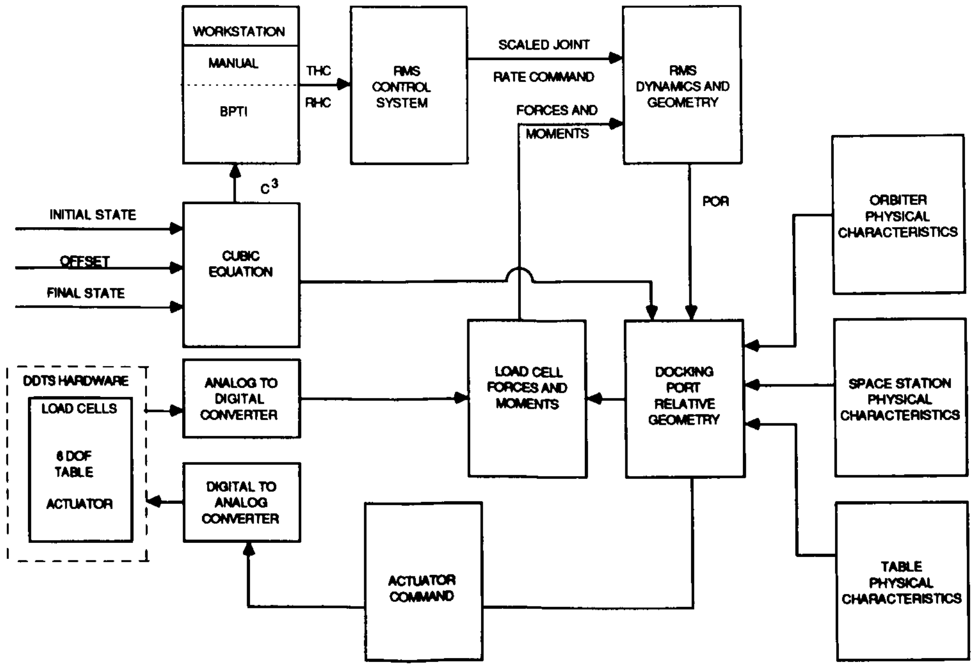
BERTHING SIMULATOR OPERATION

Figure 7 summarizes the berthing simulator operation. The hardware system is powered up to conduct a berthing simulation. The initial state, final state, total travel time, and offset condition (if any) are provided as inputs to the cubic equation model . The model calculates the desired states of the POR and the table. The cubic equation model provides the signals to the actuators (to initialize the table) and the RMS model (to initialize the POR). After the table and the POR are initialized, the real-time simulation is started.

The berthing programmed test input (BPTI) model, based on the outputs from the cubic equation model, generates rate commands to drive the POR to the desired position, attitude, and rate. The rate commands are converted to translational hand controller (THC) and rotational hand controller (RHC) digital counts for the RMS control system. Once the table and the simulated POR are synchronized, the berthing operation can be performed automatically or manually using the hand controllers.

To follow BPTI commands manually, a pilot aid of three dimensional (3 D) soccer ball shaped object with 6 DOF will be displayed on the graphics system of the RMS workstatio.ı, as shown in figure 8. If the hand controller commands are matched exactly with the BPTI commands, the soccer ball will remain static with the index ring shown in figure 8(a). Otherwise, the size of the ball will be smaller than the index ring for the positive x axis and the center of the ball will move to the right and down of the index ring center for the positive y and z axes as shown in figure 8(b). Further, the ball will spin in the y, x, and z axes for positive pitch, roll, and yaw .

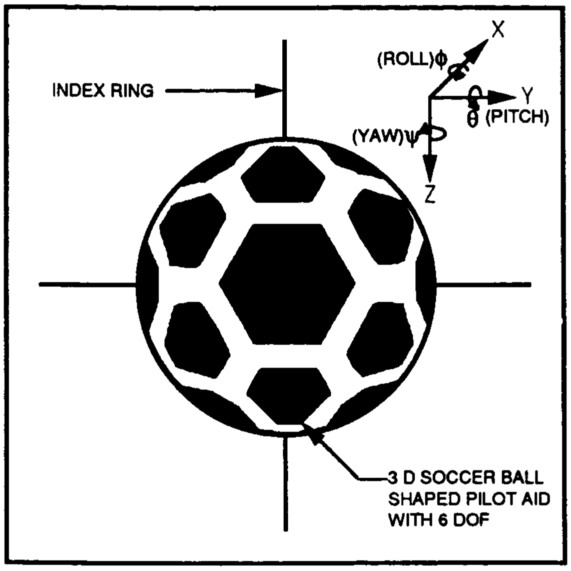
The RMS control system, shown in figure 7, calculates the scaled joint rate command. The simulated RMS control system has a capability of RMS control entry, data display for RMS joint angles,



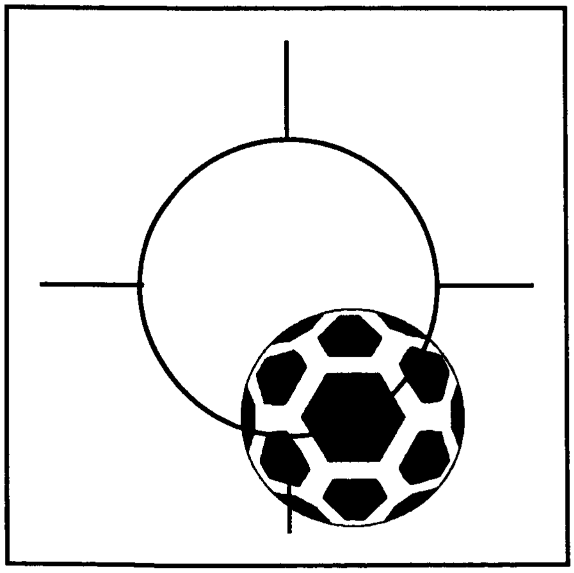
WORKSTATION
MANUAL
BPTI
THC
RHC
RMS CONTROL SYSTEM
SCALED JOINT RATE COMMAND
FORCES AND MOMENTS
RMS DYNAMICS AND GEOMETRY
C3
POR
INITIAL STATE
OFFSET
FINAL STATE
CUBIC EQUATION
ORBITER PHYSICAL CHARACTERISTICS
DDTS HARDWARE
LOAD CELLS
6 DOF TABLE
ACTUATOR
ANALOG TO DIGITAL CONVERTER
LOAD CELL FORCES AND MOMENTS
DOCKING PORT RELATIVE GEOMETRY
SPACE STATION PHYSICAL CHARACTERISTICS
DIGITAL TO ANALOG CONVERTER
ACTUATOR COMMAND
TABLE PHYSICAL CHARACTERISTICS

Figure 7. BLOCK DIAGRAM OF BERTHING SIMULATOR

rates, health monitor information etc., and the POR translational and rotational commands. The RMS dynamics and geometry models calculate the POR position and attitude. Using the orbiter, the space station, and the table physical characteristics, the docking port relative geometry model calculates the docking port relative position, attitude and rates. The actuator command model calculates the required actuator stroke length. The results are applied to the hydraulic actuators through digital to analog converters. The table moves from the initial to the final condition where the initial impact occurs. When the docking mechanisms make a contact, the load cells measure the contact forces and the results are provided to the RMS dynamics model through the analog to digital converters and the load cell forces and moments model. Actual test data begins at the initial contact of the docking mechanisms and continues through the soft latch and a pilot or a RMS operator responds to the dynamic interactions to achieve the latching.



(a) PROPERLY MATCHED HAND CONTROLLERCOMMANDS WITH BPTI COMMANDS.



(b) NEED TRANSLATIONAL (-X, -Y,-Z) HAND CONTROLLER COMMANDS TO MATCH BPTI COMMANDS.

FIGURE 8. PILOT AID DISPLAY

CONCLUSION

A real-time man-in-the-loop berthing simulator is being developed at NASA/JSC. A simulated orbiter robotic arm is used to conduct a berthing test. A dynamic initialization technic is developed to overcome the existing hardware and software constraints. The berthing test will examine the forces during contact conditions of the actual space station and the orbiter docking mechanisms. The test can be done automatically or manually to have a realistic robotic arm response at contact condition and to achieve latching. A 6 DOF pilot aid is developed for manual berthing operation. The berthing study can be extended for any two orbiting vehicles using a simulated robotic arm.

ACKNOWLEDGEMENTS

Acknowledgements are due to B.H. Strassner, Y.M. Kuo, D. Hayden, Y. Yang, A. Sylvester et al. The development of the berthing simulator is a joint effort of the Simulation Systems Branch, Systems Engineering Division, NASA/JSC, and the Simulation Development Department of Lockheed Engineering and Sciences Company.

REFERENCES

1. Strassner, Bernd, "Dynamic Docking Test System - Final Mathematical Model", NASA/JSC Internal Note 74-EG 23, July 1974.
2. Strassner, Bernd, "System Engineering Simulator On-Orbit Element Simulation Definition Document", LEMSCO - 2411, Lockheed Engineering and Management Services Company, April 1988.
3. Veerasamy, Sam, and Hayden, Don ,"DDTS Berthing Simulation Critical Design Review", NASA /JSC, Lockheed Engineering and Sciences Company, January 25,1990.

N91-20658

EVALUATION OF TELEROBOTIC SYSTEMS USING AN INSTRUMENTED TASK BOARD

Thomas C. Bryan
NASA/Marshall Space Flight Center
Mail Stop: EB24
Huntsville, Alabama 35812

John D. Carroll, Paul A. Gierow
SRS Technologies
990 Explorer Blvd.
Huntsville, Alabama 35806

ABSTRACT

An Instrumented task board has been developed at NASA Marshall Space Flight Center (MSFC). An overview of the task board design, and current development status is presented. The task board was originally developed to evaluate operator performance using the Protoflight Manipulator Arm (PFMA) at MSFC. The task board evaluates tasks for Orbital Replacement Unit (ORU), fluid connect and transfers, electrical connect/disconnect, bolt running and other basic tasks. The instrumented task board measures the 3-D forces and torques placed on the board, determines the robot arm's 3-D position relative to the task board using IR optics, and provides the information in real-time. The PFMA joint input signals can also be measured from a breakout box to evaluate the sensitivity or response of the arm operation to control commands. The data processing system provides the capability for post processing of time-history graphics and plots of the PFMA positions, the operator's actions, and the PFMA servo reactions in addition to real-time force/torque data presentation. The instrumented task board's most promising use is developing benchmarks for NASA centers for comparison and evaluation of telerobotic performance.

INTRODUCTION

Telerobotics systems are currently being developed and evaluated at a number of NASA centers, Universities and Air Force centers. The telerobotic systems are being developed to perform a wide range of on-orbit tasks. Space operations such as satellite servicing, assembly, maintainance and payload handling can be accomplished with telerobotic systems (ref. 1). An increased need for telerobotic support will emerge as the Space Station Freedom evolves. A baseline for evaluating telerobotic systems and tasks is needed to establish comparisons of the performance characteristics among the task analysis community. An instrumented task board and data acquisition system can be used to further quantify parameters used for telerobotic systems and task evaluations.

TASK BOARD DESCRIPTION

The instrumented task board system consists of an inner task frame that is instrumented to measure the forces and torques placed on the board. The inner frame accepts 19" task panels which are interchangeable. A variety of tasks can be performed simply by installing the task panel desired on the inner frame. An optical position sensing system determines the position of the telerobotic end effector relative to the task panel. A voltage breakout box is supplied to measure the joint forces on the telerobot. A user-friendly data acquisition and reduction system is integrated into the measurement systems. Figure 1 is a photograph of the task board system at MSFC with an ORU replacement task panel. Figure 2 is a photograph of a fluid transfer demonstration task board used at MSFC.

Instrumented Beam Force/Torque Sensors

A 4-point suspension system design is used to fully support the inner frame of the task board. Cantilever beams instrumented with strain gauges and signal conditioners are used in order to determine the forces and torques placed on the task board. The instrumented beam design incorporates low friction instrument linear bearings combined with a spherical bearing. The instrumented beam design results in no axial loads or torques placed on the cantilevered beams. Incorporating the low friction axial and rotational mounting methods to the task board results in a perpendicularly applied load to each instrumented beam. The unique mounting method enables the instrumented beam/strain gauges to measure components of the load on the beams allowing for force component measurements.

Optical Position Sensors

The purpose of the position monitoring system included in the instrumented task board is to record quantitative data that can be reviewed and used as an evaluation and learning tool for the development and sharpening of the operator's PFMA skills and task evaluations. The 3-D position sensing system enables the controller to know the precise coordinate or location of the end effector tool being used, in reference to the center of the task board. This system is comprised of two Hamamatus (C2399) two-dimensional position sensor systems. Each position sensor system is a compact high-resolution position sensor using a non-discrete position-sensitive detector. The non-discrete position-sensitive detector enables high-speed measurement of a moving spot with high accuracy. The position sensor is an opto-electric unit which measures the position of a single-point of infrared light focused on the sensor head. The two dimensional position sensor systems are comprised of a system controller, an infrared lens and sensor head, and a seven infrared LED cluster target.

Each two-dimensional position sensor is monitored by the Macintosh II through an analog-to-digital input/output board. The position of the target, which is mounted near the PFMA's end effector, is recorded by the sensor head. The two dimensional coordinates are transmitted as an analog input to position controller. The sensor heads are located at 90° angles from each other relative to the center of the instrumentation board, as shown in Figure 3. By placing the sensors heads 90° apart, the three dimensional envelope of coverage resembles an odd shape cube. The sensors' analog outputs are read into the computer where they are stored in a file and plotted, in real-time, on the computer monitor.

Data Acquisition System

The data acquisition system monitors and records the interaction of the PFMA operator with the instrumented task board and each postion sensor unit. The data acquisition for the system is achieved through the use of a Macintosh II, LabVIEW control software, and an analog-to-digital input/output PC board in series with an analog multiplexer board. The system's set up is shown in Figure 4.

The Macintosh II consists of a 40 mega byte hard drive, 5 mega bytes of RAM, 4 bit color video monitor card, color high-resolution monitor, and standard keyboard. The analog-to-digital input/output board (NB-MIO-16L-25) and the analog multiplexer (AMUX-64), developed by National Instruments, provides the computer with the ability to perform data aquisition on a maximum of 64 channels. The computer system is controlled with LabVIEW, a data aquisition and control graphical software developed by National Instruments. Hard copies of the raw test data and graphs of the test preformance are obtained from the Image Writer II, a dot matrix printer. This computer system provides a user friendly environment along with efficiency.

Data Acquisition System Software

LabVIEW serves as a software driver and controller for the NB-MBIO-16 and AMUX-64 hardware data acquisition boards installed in the Macintosh II. LabVIEW is a complete programming environment which allows the user to construct virtual instruments (VI's) that control and record operations that are required. The final instrument design includes integration of the sub-virtual instruments into a single virtual instrument for simultaneous data acquisition and real-time monitoring.

The building block of LabVIEW is the Virtual Instrument (VI). The Virtual Instruments in LabVIEW are the software components of the complete data acquisition and control system installed in the Macintosh II. Each VI has a front panel which specifies the inputs and outputs of the program. Figure 5 depicts the controls and indicators of the real-time measurement system. Behind the front panel in LabVIEW is a block diagram which represents the actual executable program. The block diagram represent graphical programming functions that are standard in any programming environment. Any virtual instrument that is designed can be represented as an icon that can be included in other VI's. The hierarchical structure of LabVIEW enables the user to construct complicated control and acquisition systems from combining the Virtual Instruments into one complete Virtual Instrument.

Data Acquisition System Capabilities

The data acquisition system on the Macintosh II for the arm sensor system is driven by the LabVIEW software. The data acquisition requirements of the arm sensor system include reading and storing to disk analog signals from the strain gage conditioning circuits from the task board, position sensors and current proportional voltages from the PFMA servomotors. Information from the PFMA operator via RS-232 data lines was included in the data acquisition and storage system on the Macintosh II. A summary of the operations of the data acquisition and storage system are shown in Table I.

Force Measurements
- Acquisition of the Strain Signals from the Instrumented Task Board
- Conversion of the Strain Data to Force Data
- Calculation, Reading and Real Time Graphical Presentation of the 3-D Forces Applied to the Instrumented Task Board

Position Measurements
- Acquisition of the Position Sensors' Outputs
- Calculation of the PFMA's 3-D Positive Relative to a Chosen Origin
- Recording and Graphical or Numerical Presenting the 3-D Location of the PFMA in Reference to a Chosen Origin

PFMA Servomotor Measurements
- Acquisition of the Current Proportional Voltages from the PFMA's Servomotors
- Calculation of Power Used by Each Servomotor, During Operations (if required)
- Resolve Joint Voltages to Task Analysis Primitives

PFMA Control Life Information
- Record all PFMA's Control Line Information Which is Transmitted Over an RS-232/422 Data Bus

Table I. Current Measurement Capabilities

A double buffer acquisition system is used in LabVIEW. The system allows the programmer to store information in a buffer while scanning the channels of interest on the A/D board. The buffer is then periodically read and stored to the desired output file on the computer. The data is also plotted on the screen while simultaneous data acquisitions are occurring. An external gate is also used for triggering of data acquisitions. The external gate enables data acquisitions while the gate is held in a high position. The external gate enables the board to perform a scan of the channels at a high rate but allow for a delay time between each scan. The delay time is determined by the desired acquisition rate.

The operator commands via the RS-232 data line can be read by LabVIEW using an RS-422 port read virtual instrument. The instrument reads the contents of the RS-422 buffer. The buffer size can be configured by the user. Simultaneous analog data acquisition and RS-422 buffer storage is available if desired. The RS-422 buffer read VI can be incorporated into the data acquisition VI.

CONCLUSION

Currently there are no standards for laboratory comparisons of telerobotic systems. The instrumented task board design developed to evaluate the PFMA at MSFC could be useful for establishing a benchmark tool to evaluate telerobotic systems within the NASA centers. The data taken from the instrumented task board could be used as a departure point for technical discussions among NASA centers. The task board will support standardization and further define task analysis for telerobotics. Additional task sets can be added to the task board design to include a variety of tasks that include collision avoidance, adjustable inertia crank, lighting systems and dynamic situations.

ACKNOWLEGEMENTS

This research and development is currently being supported by the Information & Electronics Laboratory, NASA Marshall Space Flight Center (Contract #NAS8-36307). Mr. Thomas C. Bryan is the technical monitor.

REFERENCES

1. Drews, Michael, "General Extravehicular and Telerobotic Task Primatives for Analysis, Design, and Integration," JPL Publication 89-XXX, Draft 1.5, Jet Propulsion Laboratory, Pasadena, CA, January, 1990.

2. Price, Charles, "Status of Benchmark Task/ Task Boards," Telerobotics Intercenter Working Group, Johnson Space Center, TX, July 25, 1989.

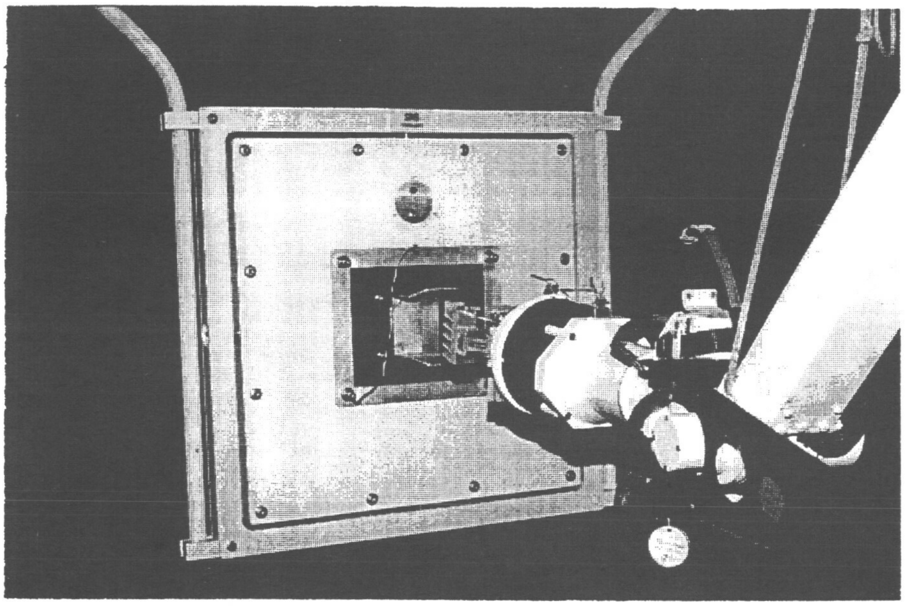


Figure 1 Instrumented Task Board

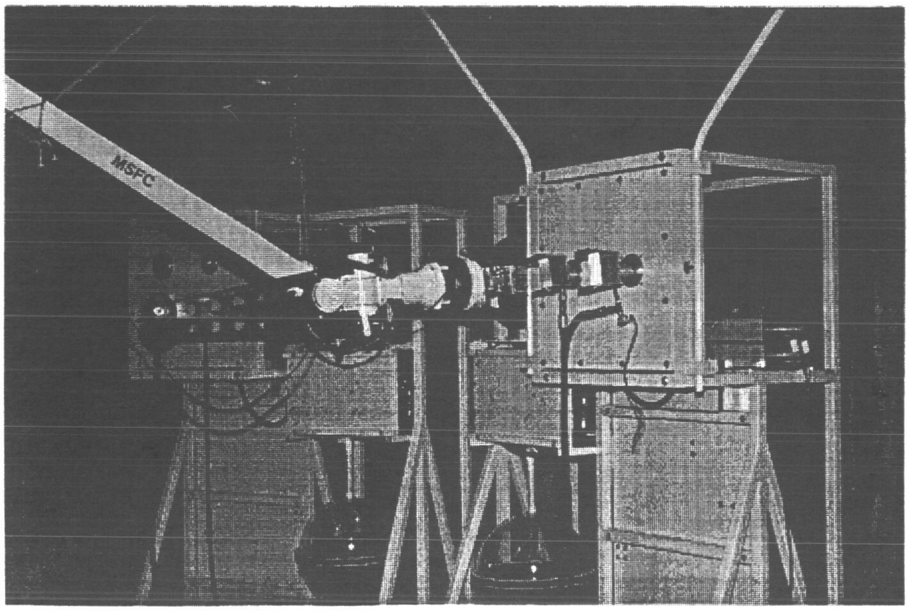


Figure 2 Fluid Transfer System

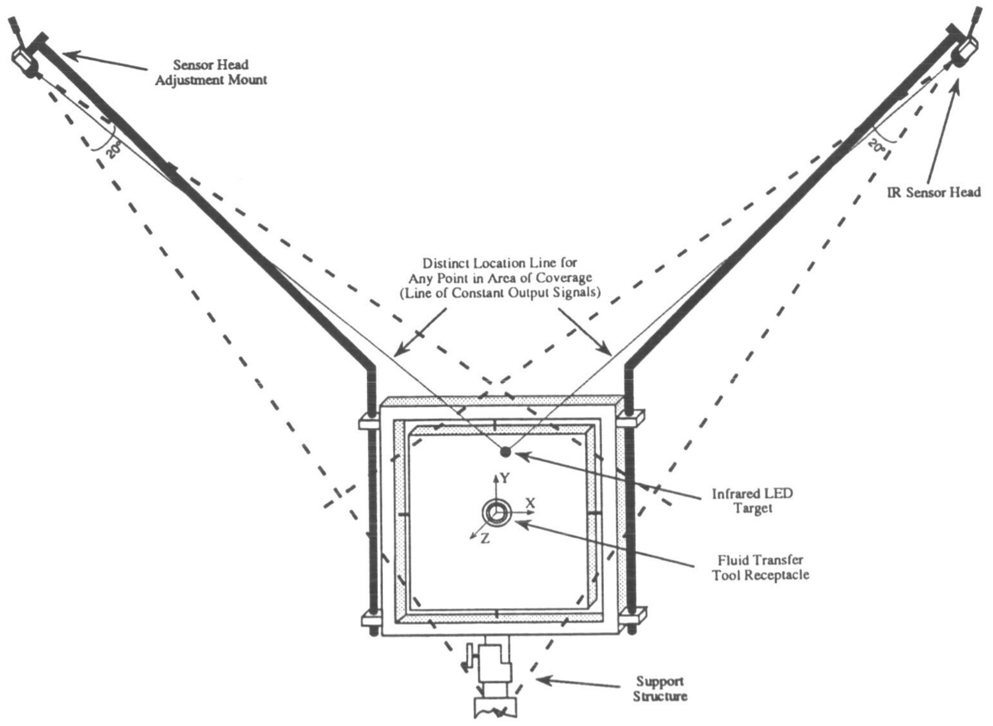

Sensor Head Adjustment Mount
20°
20°
IR Sensor Head
Distinct Location Line for Any Point in Area of Coverage (Line of Constant Output Signals)
Y
X
Z
Infrared LED Target
Fluid Transfer Tool Receptacle
Support Structure

Figure 3 IR Position Sensor System

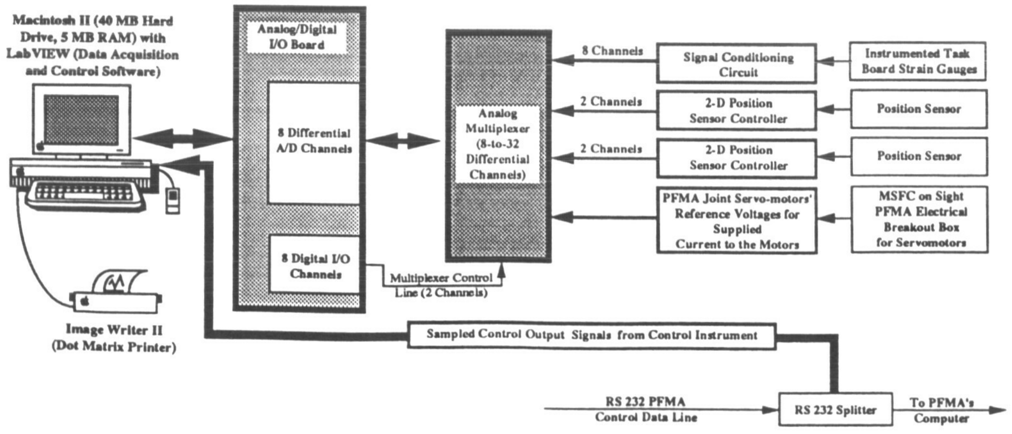

Macintosh II (40 MB Hard Drive, 5 MB RAM) with LabVIEW (Data Acquisition and Control Software)
Image Writer II (Dot Matrix Printer)
Analog/Digital I/O Board
8 Differential A/D Channels
8 Digital I/O Channels
Multiplexer Control Line (2 Channels)
Analog Multiplexer (8-to-32 Differential Channels)
8 Channels
2 Channels
2 Channels
Signal Conditioning Circuit
Instrumented Task Board Strain Gauges
2-D Position Sensor Controller
Position Sensor
2-D Position Sensor Controller
Position Sensor
PFMA Joint Servo-motors' Reference Voltages for Supplied Current to the Motors
MSFC on Sight PFMA Electrical Breakout Box for Servomotors
Sampled Control Output Signals from Control Instrument
RS 232 PFMA Control Data Line
RS 232 Splitter
To PFMA's Computer

Figure 4 Electronic Data Acquisition System

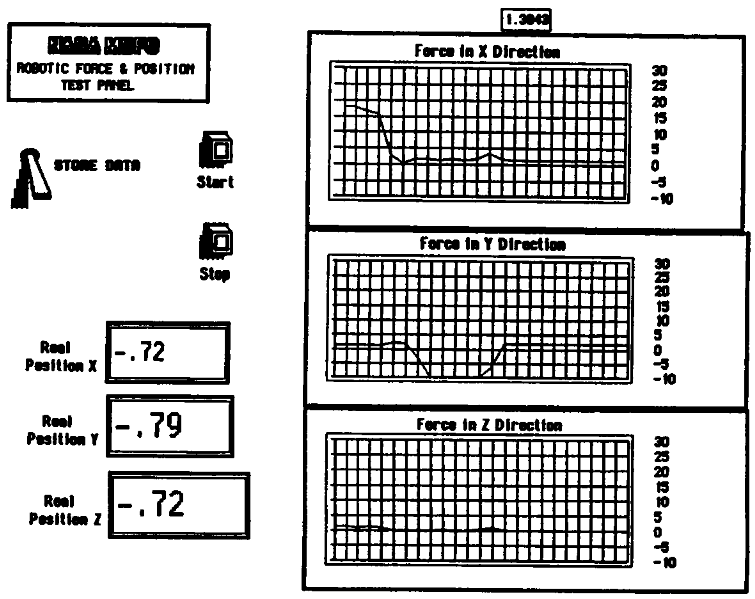


Figure 5 "Real-Time" Data Acquisition Panel

ROBOTIC SERVICING SYSTEMS

PRECEDING PAGE BLANK NOT FILMED

N91-20659

OPERATIONS WITH THE SPECIAL PURPOSE DEXTROUS MANIPULATOR ON SPACE STATION FREEDOM

B.Cox, D.Brown, M.Hiltz
Space Operations
Spar Aerospace
Weston, Ontario, Canada

ABSTRACT

SPAR Canada is actively participating in the Space Station Freedom Program by contributing the Mobile Servicing System (MSS) which will be involved in assembly, maintenance and servicing of both the Space Station and the MSS itself. Part of the MSS is the Special Purpose Dextrous Manipulator (SPDM), a two armed dextrous robot with advanced vision and manipulative capabilities. In addition to Space Station and payload servicing activities the SPDM will be designed to perform self maintenance on the MSS itself. The majority of Space Station equipment will be on orbit for the anticipated 30 year lifespan and the maintenance philosophy will be to repair by the exchange of Orbit Replacement Units or ORU's.

This paper describes the present concept, configuration and operation of the SPDM and the detailed simulations associated with the maintenance of part of the MSS. The Design Reference Mission presented in this paper is the replacement of a Joint Drive Module on the Canadian large payload manipulator, the Space Station Remote Manipulator System.

Other Design Reference Missions that have been investigated are briefly described, and future operations activity to support the definition of SPDM requirements are discussed.

ACRONYMS AND MNEMONICS

DRM	*Design Reference Mission*
EVA	*Extra Vehicular Activity*
EVR	*Extra Vehicular Robotics*
FSE	*Flight Support Equipment*
JDM	*Joint Drive Module*
LEE	*Latching End Effector*
MBS	*MSS Base System*
MSS	*Mobile Servicing System*
MMD	*MSS Maintenance Depot*
MT	*Mobile Transporter*
ORU	*Orbit Replaceable Units*
SSF	*Space Station Freedom*
SPDM	*Special Purpose Dextrous Manipulator*
SSRMS	*Space Station Remote Manipulator System*
TCM	*Tool Changeout Mechanism*

INTRODUCTION

The Special Purpose Dextrous Manipulator (SPDM) is an element of the Mobile Servicing System (MSS) which is the Canadian contribution to the International Space Station Freedom Program as defined in the Memorandum of Understanding between Canada and the USA. The MSS provides hardware for: the assembly and external maintenance of the Space Station: the servicing of attached payloads; the transport of payloads and hardware about the Station; the deployment and retrieval of free flyers; the berthing of vehicles such as the Space Shuttle; the support of Extravehicular Activity (EVA) and support of Space Station Operations including safe haven requirements. The MSS is being designed to be self maintainable, a unique and novel feature on the Space Station. The SPDM will provide the dextrous robotic capability to maintain the MSS and thus reduce use of EVA which is both hazardous and expensive.

CONCEPT

Extra Vehicular Activity (EVA) is a time consuming, expensive and potentially dangerous operation. Any viable method to reduce EVA must be given serious consideration. As evidenced by recent NASA studies (Fisher-Price, External maintenance Task Team), considerable thought is being given to the use of robotics for routine maintenance tasks on the Space Station. Repair from failure conditions and routine maintenance is performed by the changeout of Orbit Replaceable Units or ORU's. Ultimately all activities have to be achievable by EVA to cover the case of robot failure or Station resource failure (Loss of Power, Data,Video,communications). Logically, therefore, it follows that a dextrous robot should have the ability to perform tasks in confined environments where access has been determined by a suited astronaut. The SPDM must be capable of performing the following tasks, with appropriate tools: - Exchange ORU's of up to 600 Kg mass - connect/diconnect utilities - Mate/demate connectors in single or staggered rows, as little as 4cm apart in single rows - Attach/Detach interfaces - Provide lighting to the work area or EVA crew - Monitor the work area or the EVA crew by closed circuit TV - Clean surfaces - Remove/install thermal covers and blankets - Perform various inspections

The minimum and maximum reach envelopes, access restrictions and tip torques were originaly specified for the SPDM based on EVA capabilities. Robots that have to perform human-like tasks do not, however, have to have the same anthropomorphic design. For example the SPDM body can fold/unfold allowing access to difficult work volumes; an EVA astronaut could acomplish the access problem by relocating to a different foot restraint location. The DRM that follows is the present driver for the overall reach requirement of the SPDM, one of the many derived requirements that will ultimately be investigated and verified by DRM's.

CONFIGURATION *(Figure 1)*

The SPDM consists of a base section, an articulated body, two seven degree of freedom arms and a head with vision and lighting sytems.

The Base section has a length of around 1.8 metres and has one roll joint.The base supports the articulated sections of the manipulator and consists of a latching End Effector so that the SPDM can grapple and operate from a Power Data Grapple Fixture (PDGF); power, data and video resources pass across

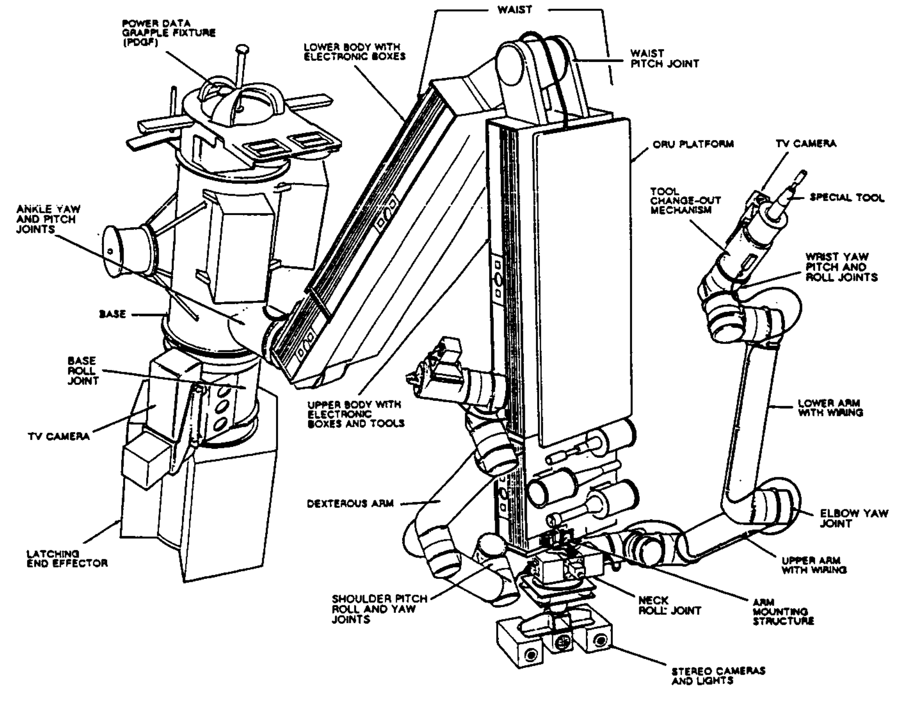

POWER DATA GRAPPLE FIXTURE (PDGF)
LOWER BODY WITH ELECTRONIC BOXES
WAIST
WAIST PITCH JOINT
ORU PLATFORM
TV CAMERA
TOOL CHANGE-OUT MECHANISM
SPECIAL TOOL
ANKLE YAW AND PITCH JOINTS
WRIST YAW PITCH AND ROLL JOINTS
BASE
BASE ROLL JOINT
UPPER BODY WITH ELECTRONIC BOXES AND TOOLS
LOWER ARM WITH WIRING
TV CAMERA
DEXTEROUS ARM
ELBOW YAW JOINT
LATCHING END EFFECTOR
UPPER ARM WITH WIRING
SHOULDER PITCH ROLL AND YAW JOINTS
NECK ROLL JOINT
ARM MOUNTING STRUCTURE
STEREO CAMERAS AND LIGHTS

FIGURE 1 SPDM CONFIGURATION

the PDGF interface. A video camera provides visual information for berthing the End Effector. PDGFs can be located at strategic worksites on the Space Station Truss, modules and on the various elements of the MSS itself. At the other end of the base is a PDGF which can be grappled by the Space Station Remote Manipulator System (SSRMS). The SPDM can therefore be carried and operated from the end of the SSRMS. Accessibility is therefore greatly enhanced as the SSRMS and its Mobile Remote Servicer system and Mobile Transporter can reach most areas on the Space Station requiring servicing activities.

The Body sections provide temporary storage for Orbit Replaceable Units (ORU's) and accommodation for the SPDM electronic processors. Tools will also be stored on the body. The body is attached to the base by a pitch and yaw joint and the central articulation is a pitch joint.

Each of the two arms are presently configured with seven degrees of freedom, each carries a Tool Changeout Mechanism (TCM) which provides interfaces between the arm itself and the servicing tools. Tools can be changed as required during operations, a video camera could also be carried as part of the TCM. Force moment sensing is to be included in the arms. Joint electronics and integrated thermal protection are also incorporated in the arms themselves.

The neck and head system allow for growth to stereo vision and it is intended to include an artificial vision function which will allow automatic tracking and grappling of ORU's, tool alignment and ORU identification. The head and neck includes both pan and tilt units and artificial lighting.

When completely folded for storage and launch the SPDM occupies a volume with dimensions of approximately 2500x876x1435 mm. The SPDM in its stowed configuration requires minimal Flight Support Equipment and is presently manifested to fly with the MSS Maintenance Depot (MMD) on flight 8 (OF-1). A current concept for SPDM launch packaging is to utilize the MMD itself or the unpressurized Logistics Sub-Carrier.

OPERATION

The SPDM is capable of operating from the end of the SSRMS, from operating locations on the MSC, MMD and from PDGF's positioned on the Space Station structure. *(Figure 2)*

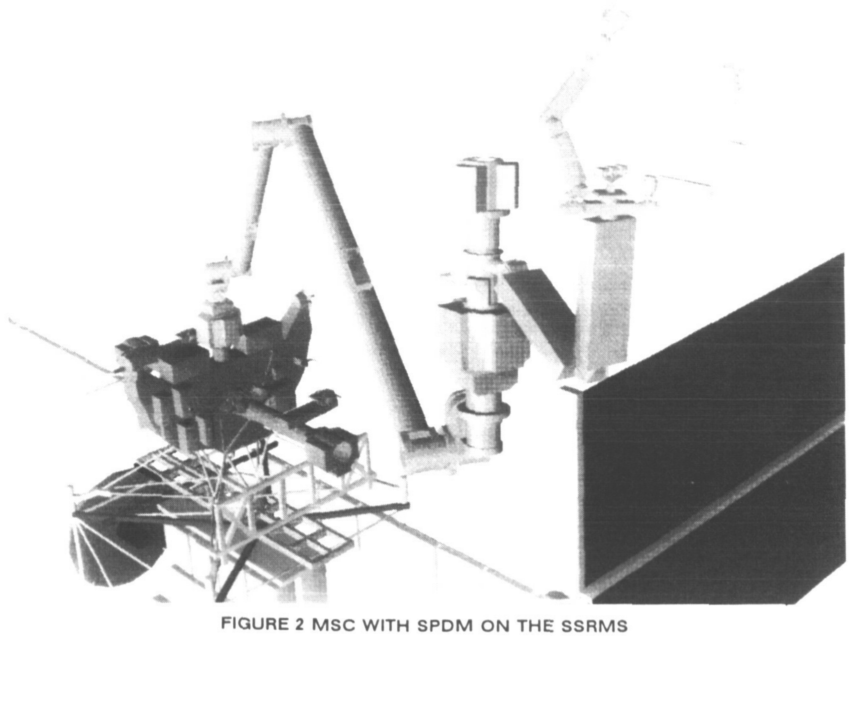


FIGURE 2 MSC WITH SPDM ON THE SSRMS

When operating from the end of the SSRMS, the SPDM PDGF on the base is grappled by the large arm and then positioned close to the area of operations. Due to the flexibility of the SSRMS, one of the SPDM arms would be used to grasp a stable piece of structure while the other arm would perform the dextrous task. An effective load path and sufficient stiffness would thus be achieved by using one arm as a stabilizer.

The MSS is designed to have several PDGF's to serve as operating locations for the SPDM. On the Mobile Remote Servicer Base System (MBS) there are two PDGF's capable of supporting SPDM operations. From these positions on opposite sides of the base the SPDM can access all the ORU's associated with the MBS and the SSRMS. The SPDM can therefore provide the robotic capability to maintain its parent system. The changeout of an SSRMS Joint Drive Module is examined as a Design Reference mission later in this paper.

PDGF's at strategic locations on the Space Station can also support SPDM operations. Present PDGF locations on the Space Station are on one of the forward nodes, the U.S. Laboratory Module and the Japanese Module.

The MSS Maintenace Depot (MMD) is at a fixed location on the truss where MSS specific spares and tools are stored and can be accessed for MSC maintenance. The Mobile Transporter can position the MSC on an adjacent truss bay to the MMD and allow the SPDM to collect tools and spares and also operate from the MMD.

One of the most versatile features of the SPDM is its ability to be carried by the SSRMS and in turn to carry a payload and also perform dextrous tasks on that payload. Such a unique capability is particularly useful during assembly operations that require positioning of payloads prior to attachment to Space Station structure or truss, the final interface manipulations being performed by the SPDM. EVA is thus reduced during the critical periods of Space Station construction. *(Figure 3)*

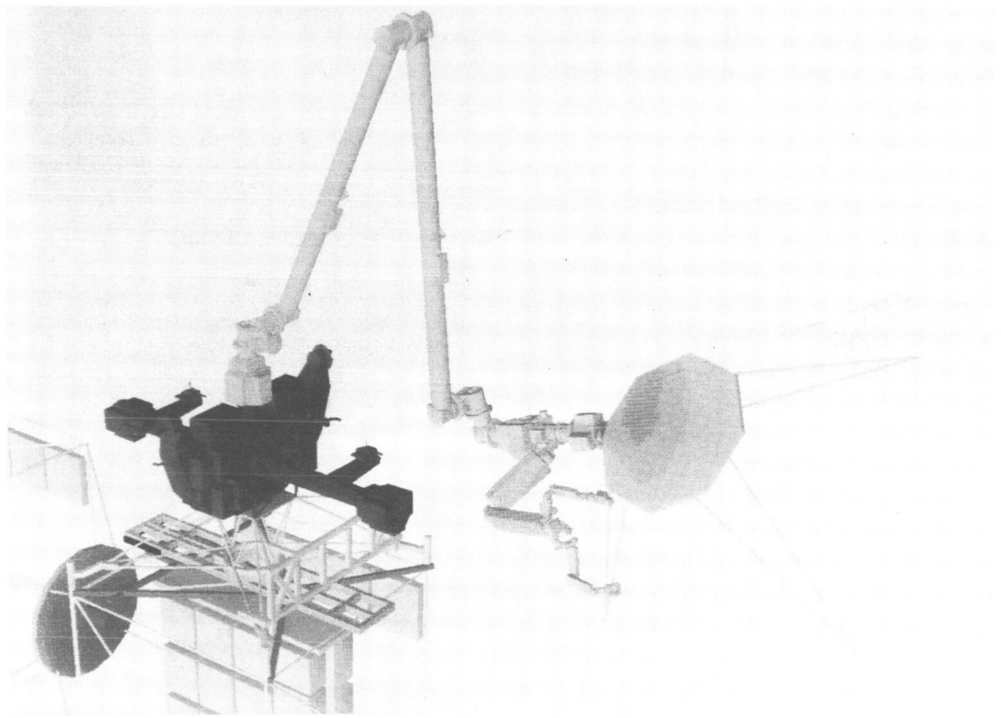


FIGURE 3 SSRMS/SPDM OPERATING ON PAYLOAD

CONTROL STRATEGY

The SPDM will be controlled teleroboticaly from a workstation inside the IVA environment. Force moment sensing and accommodation will be implemented from the start. As experience grows a number of more advanced features will be incorporated including an advanced vision system capable of direct ORU identification, auto tracking and capture and automated replacement.

There will be three modes of operation:

a) Manual Augmented mode

The human operator would input commands which would cause the motion of the arms to a particular Point Of Resolution in the task space.

b) Single Joint Mode

Individual joints of the SPDM manipulator could be commanded in rate or position mode.

c) Automatic Trajectory Mode

The control capabilities provide signals to command the manipulator along prescribed trajectories, the trajectories may be generated from stored information, POR target point coordinates or by vision system tracking signals.

DESIGN REFERENCE MISSIONS - PHILOSOPHY

The Space Operations group at Spar, in conjunction with NASA, have developed a series of Design Reference Missions which are realistic assembly and maintenance scenarios on both the Space Station and the MSS itself. One of the present principal design drivers for SPDM configuration and capabilities is the ability of the MSS to service itself. The MSS is a unique system on the Space Station, a robotic system that is largely autonomous and also capable of self maintenance. The MSS is one of the most complex systems on the Space Station and if the SPDM can maintain the MSS it will likely be able to maintain other SS equipment. Operations are therefore in a unique position to influence the requirements for the SPDM from an end user standpoint. The DRM's influence such factors as physical dimensions, joint angles and rates, working envelopes and fault tolerance architecture. As part of the studies, viewing analysis are underway to show the obstructions and lighting difficulties. Tools and ORU interfaces are also being studied.

MAINTENANCE CONCEPT

All systems on the Space Station are design to have an operational life of at least 30 years. Maintenance on orbit is achieved in the unpressurized environment by the exchange of Orbit Replacement Units. There is a trade off between designing a piece of equipment to achieve a long life without intervention for maintenance, which would be complex and costly, or keeping designs less complex and, unfortunately less reliable. The simple equation is further complicated when equipment is to be upgraded as technology advances. The principal design driver at present is the scarsity of EVA resources and a drive is being made to make ORU's suitable for robotic changeout, so called Extra Vehicular Robotics (EVR). EVR forces the design process towards commonality of robotic interfaces.

A TYPICAL SPDM DESIGN REFERENCE MISSION - SSRMS Joint Drive Module Changeout

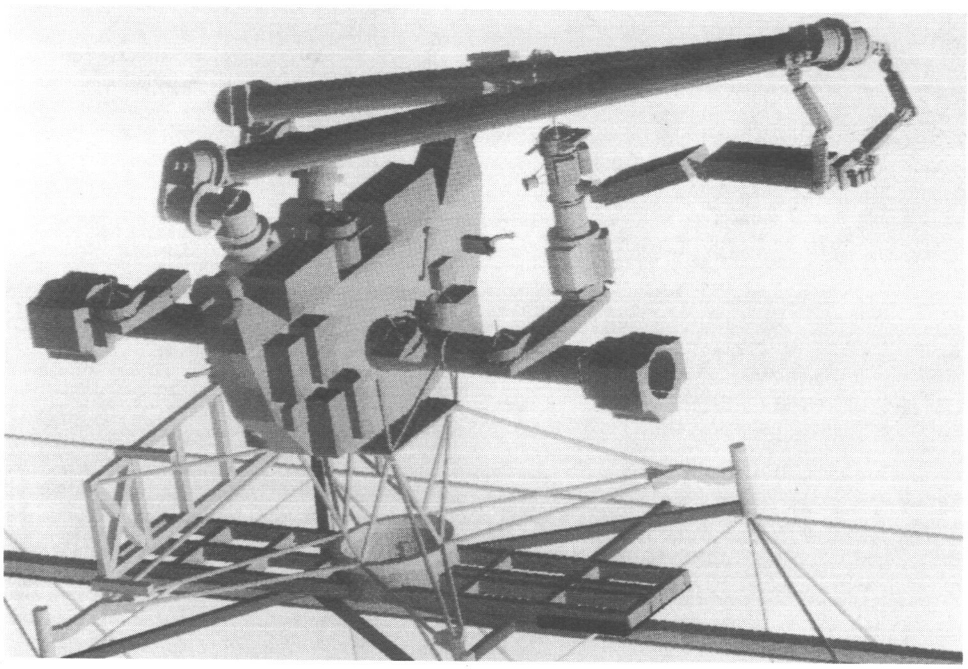
OBJECTIVE - REQUIREMENTS EXAMINED

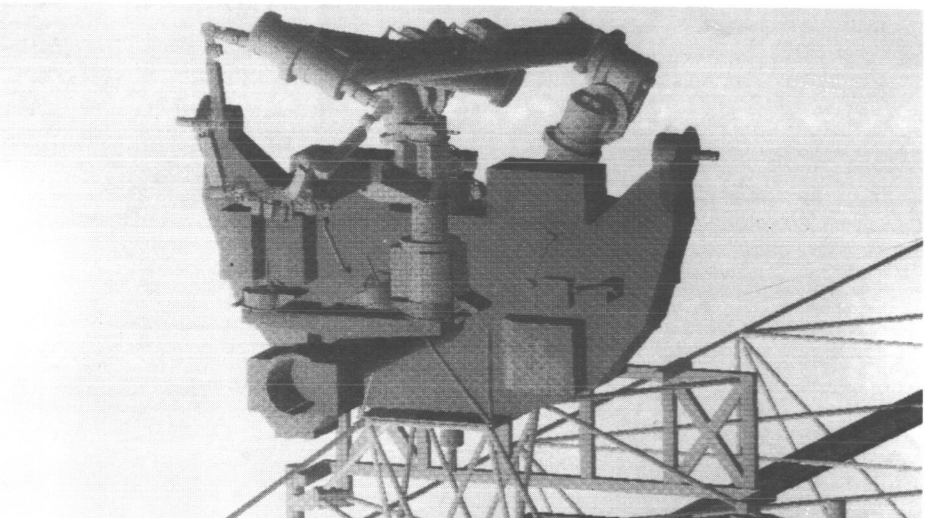
The removal and replacement of the elbow Joint Drive Module (JDM) on the SSRMS by the SPDM has been chosen as a representative example of a dextrous repair task on the MSS. The ORU changeout at the Elbow Joint demonstrates the diifculties imposed on the SPDM due to both accessibility and reach limitations. Additional requirements that are examined in this DRM are the autotrajectory and autoinsertion modes of operation as well as on board stowage of ORU's. The scenario description is given in text form together with an example of the task and sub-task breakdown.

SCENARIO DESCRIPTION

Prior to any maintenance activity the SSRMS is positioned into its maintenance configuration *(see Figures 4 & 5)* , in this position both latching End Effectors are on the MBS and the elbow joint is in the most accessible position for the SPDM. The SPDM is then unstowed from its normal PDGF.

The SPDM is placed into Auto-trajectory mode to manoeuvre close to the elbow joint and then using visual tracking and alignment the SSRMS JDM is grappled at its central interface location with one arm





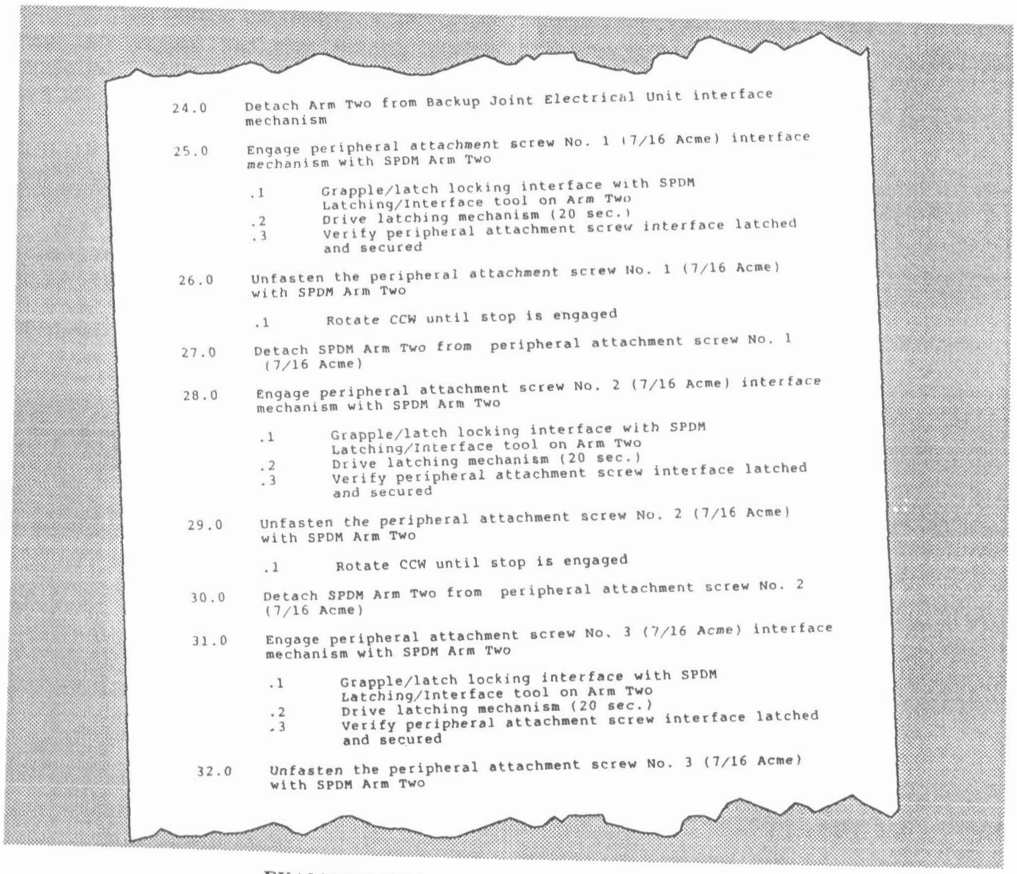
FIGURES 4&5 JDM CHANGEOUT USING SPDM ON MSC VIEW 1&2

(reference arm), this arm stabilizes the SPDM throughout the removal and replacement procedure. Arm two grapples the joint Housing Locking mechanism in order to lock it prior to removal. Arm two then locks the gear train and disconnects the primary Joint Electrical Unit umbilical. The backup JEU umbilical is diconnected in the same way after which the three peripheral attachment screws are unfastened around the JDM.

Arm two releases and reattaches to a convenient latching interface location in order to counteract the forces generated by the next step in the sequence. Arm one unfastens the central tiedown screw and withdraws the defective JDM, location coordinates are stored prior to removal. The SPDM then stows the defective JDM on its ORU stowage plate and then moves to the ORU stowage location to retrieve the replacement JDM. The defective unit is swapped with the new replacement. The SPDM returns to the worksite with the new unit and using the prestored coordinates returns to a position close to the elbow.

Arm two attaches itself to its interface point using visual alignment and rereferences the coordinates. The new JDM is now inserted using a prestored Auto insertion routine. The remainder of the assembly follows the diassembly steps in reverse order.

Following completion of the JDM installation, the SPDM withdraws to a safe standby location and a functional test routine is performed to verify that full operational capability of the SRMS has been restored.



```
24.0    Detach Arm Two from Backup Joint Electrical Unit interface
        mechanism

25.0    Engage peripheral attachment screw No. 1 (7/16 Acme) interface
        mechanism with SPDM Arm Two

        .1      Grapple/latch locking interface with SPDM
                Latching/Interface tool on Arm Two
        .2      Drive latching mechanism (20 sec.)
        .3      Verify peripheral attachment screw interface latched
                and secured

26.0    Unfasten the peripheral attachment screw No. 1 (7/16 Acme)
        with SPDM Arm Two

        .1      Rotate CCW until stop is engaged

27.0    Detach SPDM Arm Two from  peripheral attachment screw No. 1
        (7/16 Acme)

28.0    Engage peripheral attachment screw No. 2 (7/16 Acme) interface
        mechanism with SPDM Arm Two

        .1      Grapple/latch locking interface with SPDM
                Latching/Interface tool on Arm Two
        .2      Drive latching mechanism (20 sec.)
        .3      Verify peripheral attachment screw interface latched
                and secured

29.0    Unfasten the peripheral attachment screw No. 2 (7/16 Acme)
        with SPDM Arm Two

        .1      Rotate CCW until stop is engaged

30.0    Detach SPDM Arm Two from  peripheral attachment screw No. 2
        (7/16 Acme)

31.0    Engage peripheral attachment screw No. 3 (7/16 Acme) interface
        mechanism with SPDM Arm Two

        .1      Grapple/latch locking interface with SPDM
                Latching/Interface tool on Arm Two
        .2      Drive latching mechanism (20 sec.)
        .3      Verify peripheral attachment screw interface latched
                and secured

32.0    Unfasten the peripheral attachment screw No. 3 (7/16 Acme)
        with SPDM Arm Two
```

EXAMPLE OF TASK/SUBTASK BREAKDOWN

DRM ANALYSIS AND DERIVED REQUIREMENTS

From the detailed analysis and simulations there were a number of requirements that needed refinement and a number of issues.

1) The body length needed to be extended to reach the elbow.
2) Tools and Tooling requirements needed refinement for specific JDM changeout tasks.
3) IVA control and timelines were not investigated and require further study, particularly when using two arms in a coordinated fashion.
4) Viewing and lighting need further investigation.

OTHER DESIGN REFERENCE MISSIONS INVESTIGATED

In order to encompass all the SPDM requirements of routine maintenance, Assembly, Payload servicing, Space Station servicing and ORU changeout a number of other DRM's have been investigated.

DRM # 2 Japanese Exposed Facility Assembly

Outboard of the Pressurized Japanese Module is an unpressurized Exposed Facility which can be assembled using a combination of the SSRMS and SPDM. This scenario examined in detail the ability of the SPDM to perform dextrous tasks on the mating interface between the EF#1 and the JEM while supporting the Exposed Facility.

DRM # 3 Structural Interface Adapter Assembly

The Structural Interface Adapter is a platform structure that is roboticaly deployed and the interfaced to the truss structure. The platform carries payloads and equipment. The SPDM was used in this DRM to illustrate Space Station Assembly tasks.

DRM # 4 Beta Gimbal Drive Motor Module Changeout.

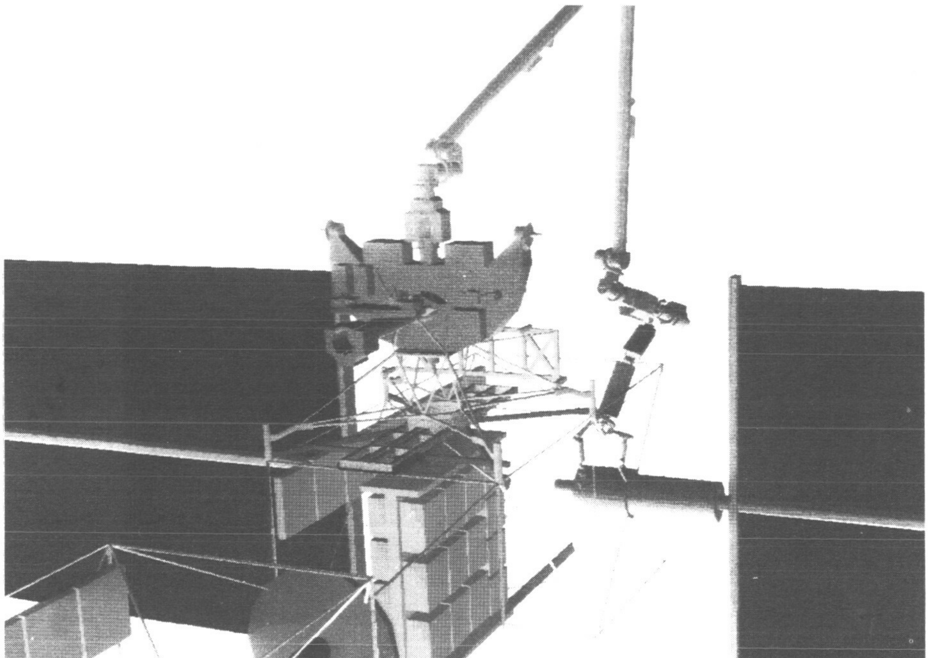


FIGURE 6 BETA GIMBAL MOTOR MODULE CHANGEOUT

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

Following a failure of part of the Solar Array equipment the SPDM was used to replace a motor module and restore the facility to full function. Dextrous operations in confined areas was the particular challenge in this DRM.

CONCLUSIONS

This paper has detailed the Operations Analysis conducted with Design Reference Mission development and the requirements that have been derived for the SPDM. In particular a DRM has been detailed which illustrates the ability of the MSS to perform self maintenance. By evaluating representative Missions, Tasks and Sub-tasks it is possible to carefully and systematically derive practical and achievable requirements and refine basic operational concepts.

FUTURE OPERATIONS

SPAR will continue operational analysis of Design Reference Missions to further refine the requirements for the SPDM. The IRIS 4D/70 GT will continue to be a valuable resource for kinematic simulations, the DENEB software is currently being enhanced so that dynamic algorithms can be implemented so that arm trajectories can be planned and evaluated.

A full scale 1G test rig called the SPDM Ground Testbed is presently being used for task assessment, tool development and ORU interface evaluation.

An Operations Simulation Facility (OSF) is presently being constructed at SPAR which will be a full scale representation of a Space Station Cupola control station with hand controllers and realistic controls. The views from the Cupola windows and workstation monitors will be simulated to represent views of the unpressurized environment. The degree of fidelity expected will allow greatly enhanced DRM operations analysis. The OSF will be of particular use in the evaluation of the Human Computer Interface and for the evaluation of Human in the Loop responses.

A Manipulator Development Simulation Facility will ultimately provide real time operator in the loop dynamic simulations of the MSS. High Fidelity mock ups of the Cupola, node, and orbiter aft flight deck will be integrated with MDSF. Ultimately, Station astronauts will be able to operate the MSS system from the Cupola workstation and examine the telerobotic responses of the SPDM in real time, and be trained in its operation.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the continued support of the Canadian Space Agency in the preparation of this paper.

REFERENCES

SPAR-SS-RD-0068
Mobile Servicing System (MSS) Requirements Definition Document (Space Segment), Issue E, March 1989.

SPAR-SS-SG-0276
Mobile Servicing System (MSS) Space Segment Contract End Item Specification (Space Station Program Phase C2-A), Issue C, March 1990.

SPAR-SS-R-0444 MSS
On-Orbit Operations Concept Document, Issue B, March 1990.

SPAR-SS-SG-0628
Special Purpose Dextrous Manipulator (SPDM) Element Specification, Issue Dr.1.0, February 1990.

SPAR-SS-DR.0630
SPDM System Concept Review Data Package Volume II, Issue A, February 1990.

Borduas, Gossain, Kong, Quittner and Shaffer, "Concept Design of the Special Pupose Dextrous Manipulator for the Space Station Mobile Servicing System", CANADIAN AERONAUTICS AND SPACE JOURNAL, December 1989.

Brimley, Burns, Cox, "MSS Space Systems Operations on Space Station Freedom", Canadian Aeronautics and Space Institute Symposium, November 1989.

N91-20660

USAF/NASA SOAR '90 Workshop
Albuquerque, N.M., 6/26-28/90

A SPACE SERVICING TELEROBOTICS TECHNOLOGY DEMONSTRATION

Edwin P. Kan and Neville I. Marzwell

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

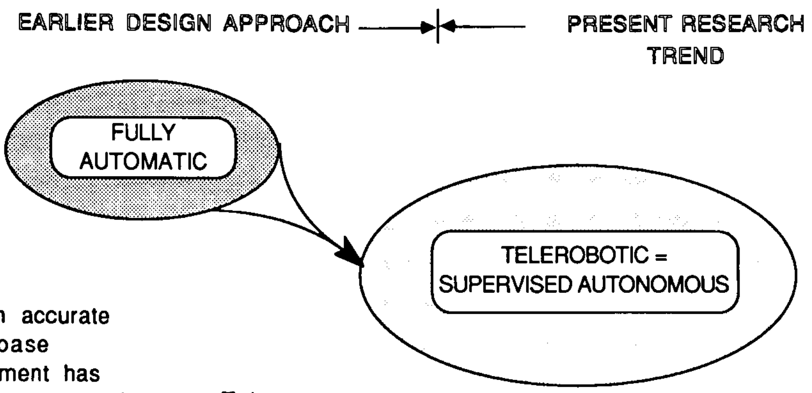
Introduction

SUPERVISED TELEROBOTIC CONTROLS PROVIDE THE KEY TO SUCCESSFUL REMOTE SERVICING, AS DEMONSTRATED IN THE TELEROBOT TESTBED OF THE JET PROPULSION LABORATORY. SUCH ADVANCED TECHNIQUES AND SYSTEMS ARE SPECIALLY APPLICABLE TO GROUND-REMOTE OPERATIONS FOR SERVICING TASKS, WHICH ARE TO BE PERFORMED REMOTELY IN SPACE AND TO BE OPERATED UNDER HUMAN SUPERVISION FROM THE GROUND. LABORATORY DEMONSTRATIONS HAVE SUCCESSFULLY PROVEN THE UTILITY OF SUCH TECHNIQUES AND SYSTEMS.

INSTRUMENTAL TO THE SUCCESS OF SUPERVISED ROBOTIC OPERATIONS ARE THE TECHNIQUES CALLED **"OBJECT DESIGNATE"** AND **"RELATIVE TARGET"**. IN ADDITION, A TECHNIQUE CALLED "UNIVERSAL CAMERA CALIBRATION" WAS ALSO APPLIED IN THE TELEROBOT TESTBED. "GENERALIZED COMPLIANT CONTROL" TECHNIQES WERE USED IN THE ROBOTIC REMOVAL AND INSERTION OPERATIONS.

THESE TECHNIQUES WERE PROVEN SUCCESSFUL IN TASK SITUATIONS WHERE PREPROGRAMMED AUTOMATION CANNOT BE ADEQUATELY EXERCISED DUE TO ERRORS, CHANGES, OMISSIONS ETC., IN THE WORKSITE DATA BASE.

Progressive Technology for Robotic Satellite Assembly and Servicing



- o Rely on accurate data base
- o Envirnoment has to be structured
- o Computer vision technique has limited performance

- o Tolerate error in data base; use 'relative target' technique
- o Can handle less structured environment
- o 'Object designate' technique uses full capability of human vision

- o Handle time delay, as in ground-remote operations
- o Expanded set of servicing demonstrations
- o Integrated system

Description of the Techniques
The Robot System & Environment

THE TASK: THE ROBOT MANIPULATOR IS TO BE COMMANDED TO

- o REMOVE THE EXISTING ORU (ORBIT REPLACEABLE UNIT) FROM ITS INSTRUMENT RACK;
- o INSERT THE ORU INTO A HOLDING RACK; and
- o FETCH A NEW ORU (SPARE), AND INSTALL IT IN THE INSTRUMENT RACK.

INFORMATION AND RESOURCES AVAILABLE:

- o DATA BASE OF WORKSITE - NOMINAL LOCATION (AND ORIENTATION) OF THE WORK SITE IS KNOWN. HOWEVER, THE ACCURACY IS CURRENTLY IMPAIRED BEYOND THE 5- 10 mm CHAMFER ACCURACY; i.e. PREPROGRAMMED AUTOMATION WILL NOT SUCCEED, GIVEN THIS ACCURACY. PRESENT TECHNIQUES APPLY EVEN IF THE DATA BASE WERE ABSENT.
- o DIMENSIONS (AND MODELS) OF ORU AND ROBOT GRIPPER ARE KNOWN TO BLUE-PRINT ACCURACY (DIGITIZED TO 1 mm ACCURACY).
- o TWO CAMERAS, MONITORS AND ASSOCIATED GRAPHICS ELECTRONICS ARE USED FOR THE PHOTOGRAMMETRIC LOCATION OF OBJECTS. COMMERCIAL GRADES ARE USED.
- o MODELS OF THE CAMERAS ARE KNOWN. PHOTOGRAMMETRIC ACCURACIES REQUIRED TO BE OF THE ORDER OF 10 cm.

Telerobotics for Space Servicing (cont.)

NEW TECHNOLOGY USED

o "OBJECT DESIGNATE" TECHNIQUE - AN OPERATOR INTERACTIVE TECHNIQUE TO UPDATE TELEROBOT DATA BASE, SPECIFICALLY OBJECT LOCATION/ORIENTATION

o "RELATIVE TARGET" TECHNIQUE - AN ALGORITHMIC APPROACH TO COMPUTE THE RELATIVE TARGETING VECTOR, USING DATA OBTAINED FROM THE SAME CAMERA VIEW OF THE ROBOT PRESENT LOCATION AND THE DESIRED ROBOT DESTINATION.

o "UNIVERSAL CAMERA CALIBRATION" MODEL - AN ALGORITHM TO DETERMINE CAMERA MODEL FOR ANY LOCATION IN ITS WORKSPACE, GIVEN ONLY ONE ABSOLUTE CALIBRATED MODEL AT ONE PREDETERMINED LOCATION.

o "GENERALIZED COMPLIANT CONTROL" TECHNIQUE - AN ADVANCED CONTROL TECHNIQUE PACKAGED TOGETHER, FOR EASY ON-LINE SEQUENCING OF CONTROL ACTIONS REQUIRED OF ROBOTIC SERVICING.

o "GUARDED MOVE" AND "MOVE-TO-TOUCH" TECHNIQUE - ADVANCED CONTROL PRIMITIVES, USED IN CONJUNCTION WITH THE COMPLIANT CONTROL PRIMITIVE.

"Object Designate" Technique -

PURPOSE: TO DETERMINE THE LOCATION (AND ORIENTATION) OF AN OBJECT IN THE WORKSPACE, WHEN THE DATA BASE ON THIS OBJECT IS INACCURATE OR ABSENT.

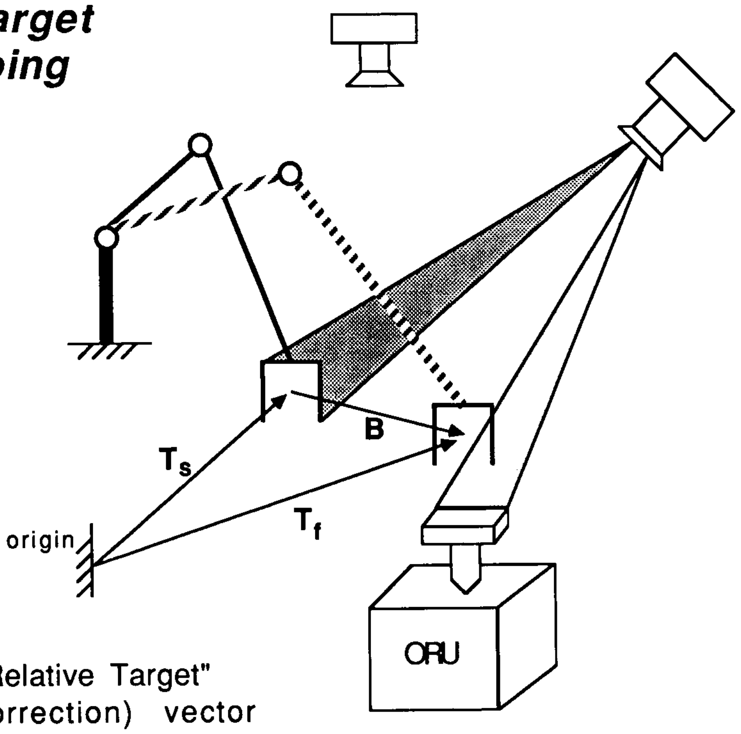
HOW IT IS PERFORMED: VIA A HUMAN-MACHINE INTERACTIVE GRAPHIC OVERLAY PROCESS, USING THE MOUSE TO DESIGNATE AND CLIKE POINTS ON VIDEO MONITORS, THE OPERATOR:

i. LOCATES THE OBJECT IN A TWO VIDEO IMAGES;
ii. DESIGNATES SEVERAL VERTICES OF THE OBJECT IN THE VIDEO IMAGES;
iii. ASSOCIATES THESE VERTICES TO THE OBJECT MODEL, AS OVERLAID ON THE VIDEO;
iv. EVALUATE THE GOODNESS OF FIT; COMPUTE AND UPDATE OBJECT DATA BASE.

RESULTS OF OBJECT DESIGNATE PROCESS: THE OBJECT DATA BASE IS THEN UPDATED (if previous readings are inaccurate), OR GENERATED (if it is absent to start with); TO THE ACCURACY PERMITTED BY THIS GRAPHICS AND HUMAN VISION PROCESS. ACCURACIES EASILY ATTAINABLE WITH COMMERCIAL PRODUCTS ARE REQUIRED TO BE WITHIN 10 cm.

IF OBJECT DESIGNATE PROVIDES ACCURACY OF WITHIN 5-10 mm., AUTOMATIC ROBOT ACTIONS CAN PROCEED. SUCH ACCURACY IS ACHIEVABLE WITH CALIBRATED CAMERA MODELS. OTHERWISE, USE RELATIVE TARGET TECHNIQUE.

Relative Target robot servoing technique



B = "Relative Target" (correction) vector

"Relative Target" Technique

<u>PURPOSE:</u> TO FIND THE RELATIVE, I.E. CORRECTION VECTOR, SO THAT THE ROBOT WILL BE DIRECTED FROM ITS APPROACH POINT TO ITS DESTINATION TARGET POINT; ACCURACY ATTAINED AT THE TARGET POINT WILL BE WITHIN 5-10 mm, i.e. WITHIN CHAMFER TOLERANCES. THIS TECHNIQUE EMPHASIZES ON A RELATIVE VECTOR, INSTEAD OF AN ABSOLUTE VECTOR.

<u>HOW IT IS PERFORMED</u>: THIS RELATIVE TARGET PROCESS IS OPERATED IN CONJUNCTION WITH THE OBJECT DESIGNATE PROCESS:

i. OBJECT DESIGNATE THE APPROACH POINT, i.e. present robot gripper location;
ii. OBJECT DESIGNATE THE TARGET POINT, i.e. ORU grapple lug location (minus a safe approach vector);
iii. COMPUTE RELATIVE VECTOR (a software process);
iv. SEND RELATIVE VECTOR TO ROBOT CONTROLLER FOR SUBSEQUENT MOTION.

<u>RESULTS OF RELATIVE TARGET PROCESS:</u> ROBOT WILL BE DIRECTED ACCURATELY TO TARGET POINT, BEYOND WHICH AUTOMATIC ORU REMOVAL/INSERTION ACTIONS CAN BE COMMANDED. ACCURACIES ACHIEVED ARE WITHIN 5-10 mm. (COMPLIANT ROBOT MOTIONS ARE NORMALLY EXERCISED.)

Generalized Compliant Motion Primitive

<u>PURPOSE</u>: PROVIDE THE SPECIFICATION OF ROBOT MOTION EXECUTION COMMAND VIA PARAMETRIZATION. PROVIDE A SIMPLE, LOW BANDWIDTH, AND UNIFIED INTERFACE BETWEEN THE PLANNER AND THE ROBOT CONTROLLER. (PLANNER CAN BE AN ARTIFICIAL INTELLIGENCE PLANNER OF SIMPLY THE HUMAN OPERATOR.)

THE GENERALIZED COMPLIANT MOTION PRIMITIVES, ALSO KNOWN AS MOTION MACROS, UNIFY THE SPECIFICATION OF MOTIONS SUCH AS (i) FORCE-POSITION COMPLIANT CONTROL; (ii) GUARDED MOTION; AND (iii) MOVE-TO-TOUCH ROBOT MOTIONS. ALL FORSEEABLE MOTIONS TYPICAL IN A ROBOT OPERATION SCENARIO CAN BE SPECIFIED BY THIS GENERALIZED PRIMITIVE/MACRO, INCLUDING THE FOLLOWING:

- PIN INSERTION / REMOVAL
- DOOR OPENING
- CRANK TURNING
- CONTOUR FOLLOWING
- PUSHING
- SLIDING
- LEVELING GRIPPERS ON GRAPPLE LUGS
- etc.

ORU Removal Sequence

STEP 1 STAGE ROBOT GRIPPER TO VICINITY (e.g. 25 cm above) OF THE ORU GRAPPLE LUG.

STEP 2 EXAMINE WHETHER GRIPPER APPEARS TO BE ABOVE ORU GRAPPLE LUG.
(i) IF YES, MOVE GRIPPER TO FINAL APPROACH POINT (5 cm above), AND PROCEED TO STEP 6.
(ii) IF NO, GO TO STEP 3.

STEP 3 INITIATE 'OBJECT DESIGNATE' OPERATION OF ROBOT GRIPPER
(a) ACQUIRE CAMERA ARM POINTING POSITION
(b) COMPUTE CURRENT CAMERA '<u>cahv</u>' MODEL PARAMETERS
(c) SEND GRIPPER MODEL AND CAMERA MODEL TO OCS
(d) PERFORM MULTI-POINT 2-VIEW DESIGNATION
(e) PERFORM LEAST SQUARES FIT TO OBTAIN GRIPPER T6
(f) TRANSMIT NEW GRIPPER T6 TO 'RELATIVE TARGET' PROCESS

STEP 4 REPEAT STEP 3 FOR ORU GRAPPLE LUG

STEP 5 COMPUTE 'RELATIVE TARGET' VECTOR FOR GRIPPER FINAL APPROACH POINT (5 cm above grapple lug)

STEP 6 EXAMINE VECTOR. IF OK, PROCEED; OTHERWISE GO BACK TO STEP 3.

(to be continued)

ORU Removal Sequence (cont.)

STEP 7 INITIATE MOVE_TO_TOUCH COMPLIANT CONTROL PRIMITIVE. (Use this primitive with the appropriate parameters, including gripper model, payload model, maximum safe travel distance, encounter force/torque, backoff force/torque, coordinate systems, robot specifications etc. In demo, this is mtouch_tri_z100_r macro-primitive.)

STEP 8 IF SUCCESSFUL, PROCEED; OTHERWISE ABORT, OR GO BACK TO STEP 3

STEP 9 INITIATE COMPLIANT_GRASP COMPLIANT CONTROL PRIMITIVE. (Use this primitive with the appropriate parameters, including maximum push force, safety limits, application coordinate systems, gripper close command etc. In demo, this is grasp_close_fz10_r macro primitive.)

STEP 10 IF SUCCESSFUL, PROCEED; OTHERWISE ABORT, OR GO BACK TO STEP 3.

STEP 11 INITIATE COMPLIANT_REMOVE COMPLIANT CONTROL PRIMITIVE TO REMOVE ORU. (Use this primitive with the appropriate parameters, including distance to be traveled, maximum force/torque limits, tool frames etc. In demo, this is remove_electr_mainNegx_r macro primitive.)

STEP 12 IF SUCCESSFUL, TASK IS DONE; OTHERWISE ABORT OR REPEAT WITH DIFFERENT PARAMETERS.

Conclusion

DEMONSTRATED MERITS OF THIS TELEROBOTIC APPROACH

- o *TOLERANCE* TO LESS STRUCTURED ENVIRONMENT - INCORRECT DATA BASE CAN BE INTERACTIVELY UPDATED BY THE "OBJECT DESIGNATE" TECHNIQUE
- o *ROBUST* DESIGN - DEGRADATION OF ROBOT ARM MODELING PARAMETERS CAN BE TOLERATED, USING THE "RELATIVE TARGET" TECHNIQUE
- o *ROBUST* DESIGN - ABSOLUTE CAMERA CALIBRATION OVER THE ENTIRE VOLUME IS NO LONGER NECESSARY; ONLY ONE SET OF CALIBRATED DATA (AT ANY ONE POINT IN THE WORK VOLUME) IS REQUIRED.
- o *FLEXIBILITY* - USING THE GENERALIZED ROBOT COMMAND MACROS, A COMMAND SEQUENCE CAN BE EASILY TAILORED TO EACH SERVICING SCENERIO
- o *SUPERVISED AUTONOMY* - THIS CAN OVERCOME CONTROL BARRIERS SUCH AS THOSE FROM TIME DELAYS
- o SUITABLE FOR *GROUND-REMOTE OPERATIONS* - THIS APPROACH IS BEST SUITED FOR GROUND-REMOTE and/or STATION-REMOTE OPERATIONS.

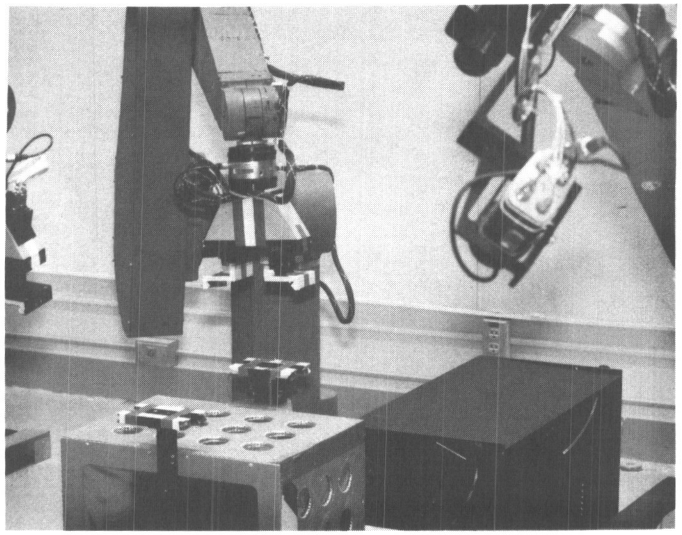
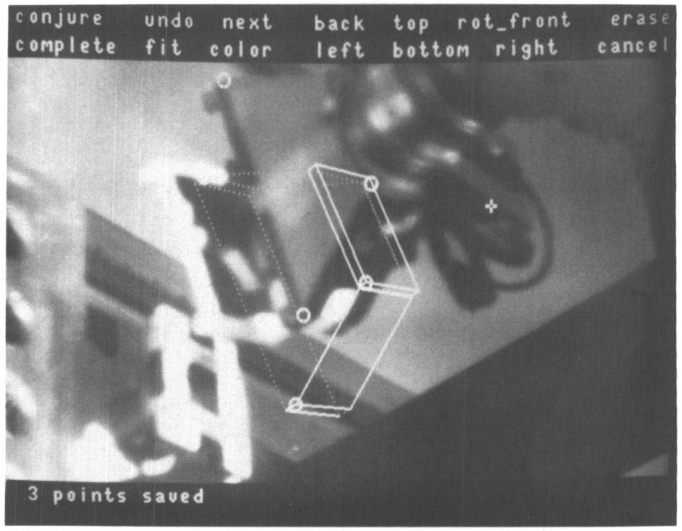
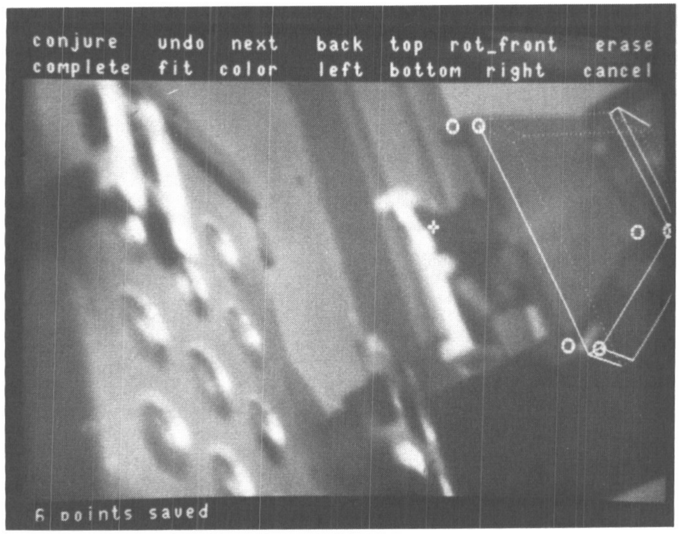


Figure 1 ROBOT WORKSITE AT THE JPL TELEROBOT TESTBED - ROBOT ARM STAGED OVER ORU; CAMERAS ON CAMERA ARM


conjure undo next back top rot_front erase
complete fit color left bottom right cancel

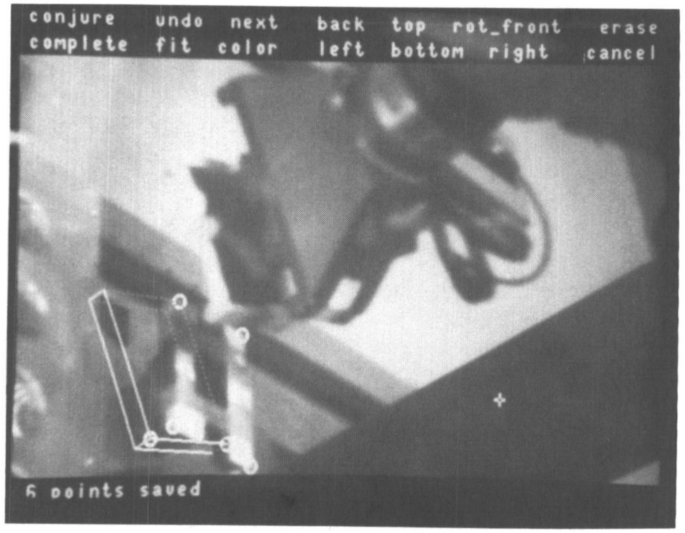
3 points saved

FIGURE 2 "OBJECT DESIGNATE" ROBOT GRIPPER - WIRE FRAME MODEL OVERLAID ON VIDEO; RIGHT CAMERA VIEW


conjure undo next back top rot_front erase
complete fit color left bottom right cancel

6 points saved

FIGURE 3 "OBJECT DESIGNATE" ROBOT GRIPPER - WIRE FRAME MODEL OVERLAID ON VIDEO; LEFT CAMERA VIEW


conjure undo next back top rot_front erase
complete fit color left bottom right cancel

6 points saved

FIGURE 4 "OBJECT DESIGNATE" ORU GRAPPLE LUG - WIRE FRAME MODEL OVERLAID ON VIDEO, RIGHT CAMERA VIEW

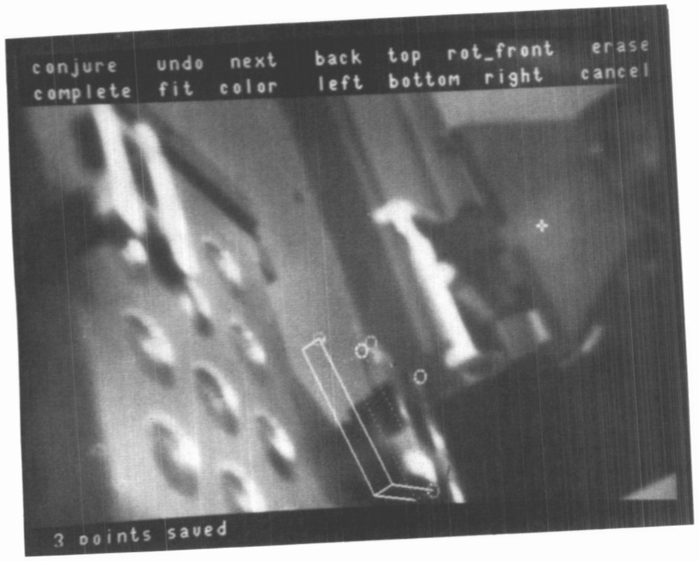

conjure undo next back top rot_front erase
complete fit color left bottom right cancel
3 points saved

Figure 5 "OBJECT DESIGNATE" ROBOT GRIPPER - WIRE FRAME MODEL OVERLAID ON VIDEO; LEFT CAMERA VIEW

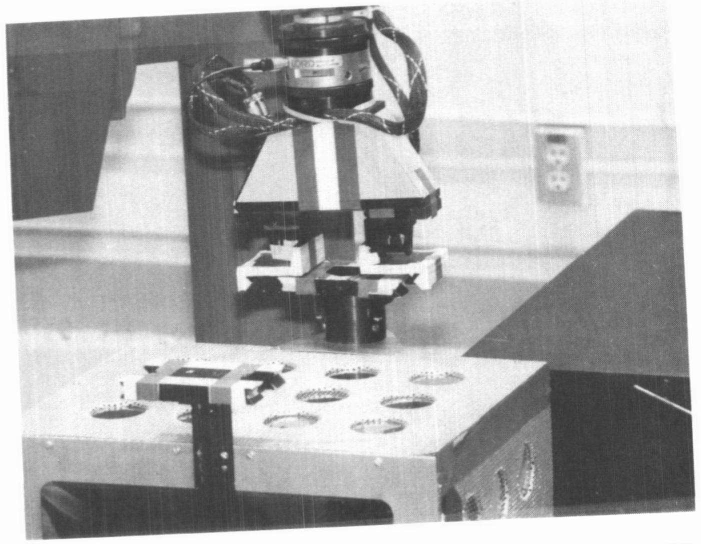


FIGURE 6 "RELATIVE TARGET" VECTORING OF ROBOT GRIPPER TO ORU, THEN "MOVE_TO_TOUCH" COMMANDING OF ROBOT

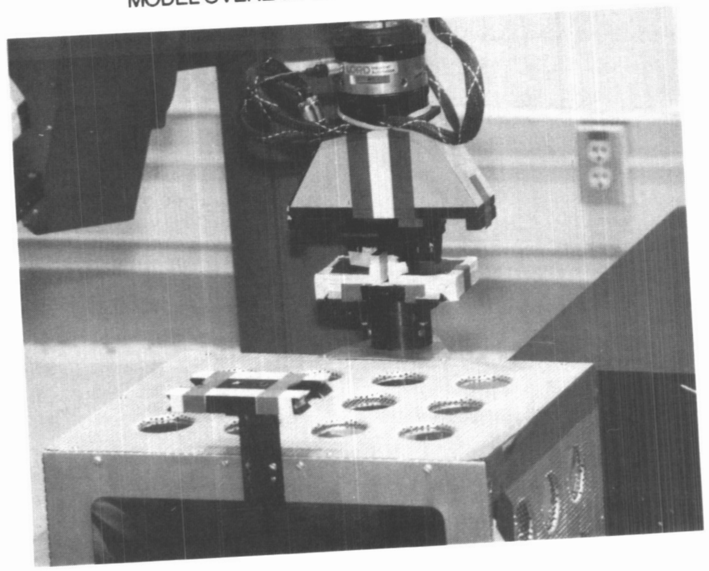


FIGURE 7 RESULT OF "COMPLIANT_GRASP" COMMAND - A GENERALIZED COMPLIANT CONTROL MACRO COMMAND

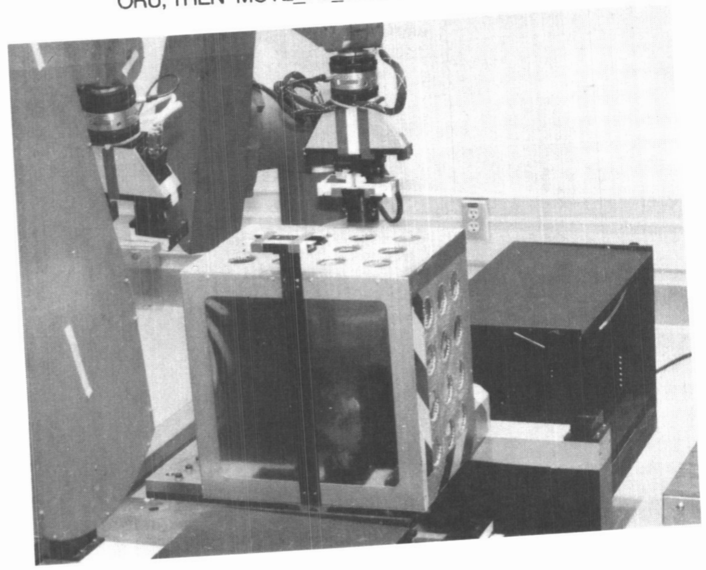


FIGURE 8 RESULT OF "COMPLIANT_ORU_REMOVE" COMMAND - A GENERALIZED COMPLIANT CONTROL MACRO COMMAND

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

N91-20661

THE DEVELOPMENT TEST FLIGHT of the FLIGHT TELEROBOTIC SERVICER

J. Andary and P. Spidaliere
Goddard Space Flight Center
Greenbelt, MD 20771

R. Sosnay
Martin Marietta Astronautics Group
Denver, CO 80201

ABSTRACT

The Development Test Flight (DTF-1) is the first of two shuttle flights to test operations of the Flight Telerobotic Servicer (FTS) in space and to demonstrate its capabilities in performing tasks for Space Station Freedom. The DTF-1 system, which Martin Marietta Astronautics Group is designing and building for the Goddard Space Flight Center, will be flown in December, 1991, as an attached payload on the shuttle. This article discusses the design of the DTF-1 system, the tests to be performed, and the data to be gathered.

INTRODUCTION

The FTS project was formed in 1986 as part of the space station work package 3 at Goddard Space Flight Center to develop a telerobotic device for performing assembly, maintenance, servicing and inspection tasks on the space station [1,2]. Before the final version is launched on one of the early space station assembly flights, there will be two early shuttle test flights: the Development Test Flight (DTF-1), now scheduled for launch in late 1991, and the Demonstration Test Flight (DTF-2), scheduled for launch in late 1993. The preliminary design review for DTF-1 was held July, 1989, and the critical design review is scheduled for September, 1990.

From the initial beginnings of the FTS project, it was recognized that an early development flight would be necessary in order to validate the FTS hardware design and to gather critical engineering test data for ground evaluation and calibration of the ground testing facilities. The DTF-1 was designed under the ground rule that the standard shuttle allocations for a quarter-bay payload be used where possible in order to contain costs and to maintain schedule.

The DTF-1 configuration consists of a payload bay element and an aft flight deck element. The payload bay element [figure 1] contains the telerobot with a single manipulator and all associated equipment, such as end-of-arm tooling, cameras and lights, task elements, and support avionics. The payload bay element primary support structure is a Multi-Purpose Experiment Support Structure (MPESS) which mechanically attaches the DTF-1 hardware to the shuttle.

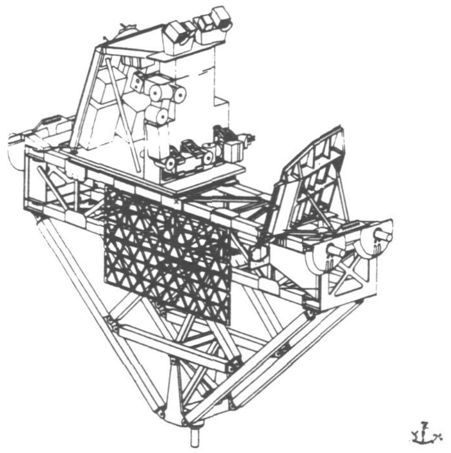


Figure 1. Configuration of Payload Bay Element for the Development Test Flight

The aft flight deck element consists of the control panels, the hand controller, and the crew restraint system, which are removable and stowable in the mid deck locker. In addition there are permanently mounted electronic boxes in the L-10 and L-11 panels.

MISSION OVERVIEW

MISSION OBJECTIVES

The DTF-1 flight hardware, flight software, task elements and mission timeline have been designed to meet the following mission objectives:

1. Evaluate the telerobot manipulator design approach
2. Evaluate the shuttle workstation design approach
3. Correlate system performance in space with ground simulation and analyses
4. Evaluate human-machine interface and operator fatigue
5. Demonstrate telerobot potential capabilities

A sequence of tasks will be performed during the 16-hour mission timeline to support these objectives. After an initial system familiarization demonstration, the remainder of the mission activities are grouped into two main categories: performance verification tasks and capabilities demonstration tasks.

The performance verification tasks include a fine positioning test, operational envelope evaluation, manipulator dynamics model verification, manipulator non-linear model verification, and a thermal transient response test. The capabilities demonstration tasks include the peg-in-hole demonstration, contour board tracking, connector demate and mate, truss strut element removal and replacement, and removable mass manipulator loading test. These tasks will be described in the operations section of this paper.

PAYLOAD BAY EQUIPMENT

The telerobot configuration used for DTF-1 consists of a single manipulator and a modified telerobot body which is large enough to support the manipulator and mount the applicable subsystems. The telerobot sits on the MPESS pallet in the cargo bay facing the task panel. The manipulator is secured for launch and landing by four caging mechanisms, two on the lower arm link and two on the wrist. A similar caging mechanism secures the removable mass on the task panel.

The caging mechanism system is one-fault tolerant against the inadvertent release of the manipulator or removable mass under all loading conditions. The caging mechanism system is also one-fault tolerant to safe the telerobot for landing.

Manipulator

The DTF-1 manipulator [figure 2] is a 7-degree-of-freedom (DOF) manipulator, approximately 5.5 feet long from the shoulder to the toolplate. The manipulator can produce 20 pounds of force and 20 foot-pounds of torque at the tool plate anywhere in the work envelope. The shoulder roll, yaw and pitch actuators are of similar design and each produce a peak torque of 118 ft.lbs. The elbow pitch actuator is capable of 61 ft.lbs. peak torque. The wrist yaw, pitch and roll actuators are of similar design and each produce a peak torque of 24 ft.lbs.

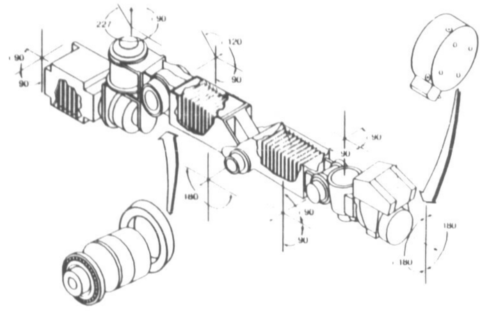


Figure 2. FTS Manipulator

Figure 3 shows the Martin Marietta engineering development wrist pitch/yaw joints. The joint actuators consist of a primary brushless dc motor; harmonic drive transmission; redundant output joint torque sensors; redundant, resolver-based output position sensors; fail-safe brakes; and housings and bearings that carry structural loads. The actuators also incorporate a secondary brushless motor, which permits independent control and safing of the manipulator through a hardwire control system, which bypasses all the computers and allows the operator to drive a single joint at a time from the workstation.

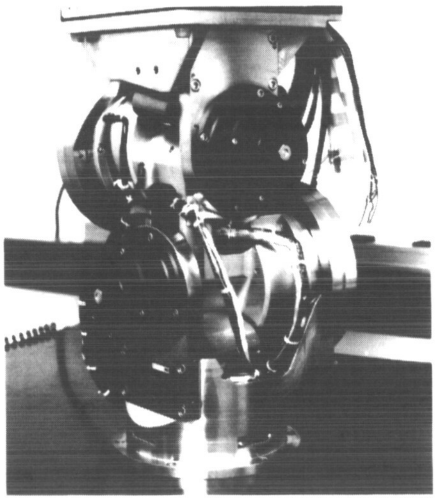


Figure 3. Martin Marietta Engineering Development Model of the Wrist Pitch/Yaw Manipulator Joints

The actuator brakes can be manually or robotically engaged and disengaged to allow backdriving of joints. This backdriving permits an astronaut on extravehicular activity (EVA) or, for future missions, another robot, to stow the manipulator in the event that the normal stowing procedure is disabled by a failure. All the manipulator electronics are contained within the manipulator.

The manipulator, under active control, will be accurate to only +/- 1 inch translation and +/- 3 degrees orientation. The repeatability of the DTF-1 manipulator in a thermally constant environment will be less than +/- 0.005 inches translation and +/- 0.05 degrees orientation. The incremental motion or resolution of the manipulators is less than 0.001 inches translation and less than 0.01 degrees orientation. All measurements are referenced at the center of the tool plate.

The manipulator includes the camera assembly mounted on the wrist roll assembly to allow the operator to closely view the end effector and tool and the objects to be manipulated. The camera will be discussed further in the vision subsystem section.

A redundant force/torque transducer (FTT) is mounted on the end of the manipulator. The FTT contains two independent strain gauge elements and associated electronics for measuring the forces and torques produced at the tool plate. The output of the FTT consists of two sets of six differential, analog signals which roughly correspond to the six independent components of the force and torque vectors. The analog outputs are provided to two independent controller electronics which digitize and calibrate the signals to generate a digital representation of the force and torque vectors at the FTT.

The manipulator tool plate allows power, data and video to be passed through to the end effector. The DTF-1 will fly a single end effector with a simulated end effector changeout mechanism (EECM). The EECM will be used in future missions to permit replacement of end effectors and tools. The DIF-1 end effector will be a single parallel jaw gripper which will perform all the mission tasks. The end effector consists of a brushless dc motor; pancake harmonic drive transmission; redundant finger position and output torque sensors; redundant, fail-safe brakes and associated gear reduction; sensor amplifiers; wiring; and connectors. The fingers are integral to the end effector and interface to the tasks using a dedicated interface. The end effector provides a peak gripping force of 50 lbs. over a 4-inch gripping range.

Task Panel

The DTF-1 task panel [figure 4] is mounted on the MPESS in front of the manipulator. The task panel holds the task elements which are designed to test the FTS's capability to meet the mission requirements. They consist of a peg-in- hole pattern, a contour board, a space station truss node, a space station fluid connector, and a removable mass.

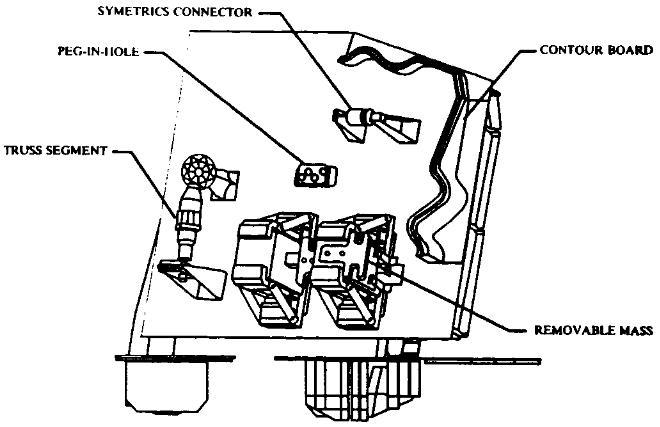


Figure 4. Task Panel

Data Management and Processing Subsystem (DMPS)

The DMPS is a distributed system of computers, controllers, data and video recorders, and hardwire control system that supports the DTF-1 software and system architectures. These electronics are connected through MIL-STD-1553b buses for data transfer.

The DTF-1 configuration consists of one telerobot control computer and eight controllers. The telerobot control computer is a prototype of the Space Station Freedom data processor. It contains two 20 MHz CPUs with 4 MBytes memory. The telerobot control computer is the primary control processor. In the forward loop, it takes commands from the hand controller, computes the inverse kinematics, and produces the required motion at the manipulator. In the return loop, the control computer receives inputs from the FTT, computes force feedback commands for the hand controller, and performs boundary management/touch control and housekeeping safety checks.

The eight controllers include the display assembly controller and hand controller drive electronics located in the workstation, three controllers located in the manipulator, the telerobot redundant controller located on the MPESS, the power module controller, and the payload bay controller. The display assembly controller and the hand controller drive electronics provide the operator interface through the control and display panel and hand controller respectively. The manipulator controllers perform the position, rate, and torque servo loop calculations and drive the joint actuators and gripper. The telerobot redundant controller collects accelerometer data and performs backup boundary management/touch control checks and housekeeping checks. The power module controller monitors power subsystem voltages and currents and controls power switching. The payload bay controller provides uplink and downlink through the shuttle and controls the head cameras and caging mechanisms. A typical controller consists of a CPU board with an 80386, 20 MHz processor and 256 MBytes memory, a 22 channel analog acquisition board, an input/output board, and a power supply board.

The data recorders are used to initialize the software and store on-orbit engineering data. Initialization takes place through the MIL-STD-1553b buses. Software load files are pre-recorded before launch.

Software files may be uploaded or downloaded while on orbit through the shuttle multiplexer/demultiplexer and payload data interleaver respectively.

The software architectural design defines an organization of software components corresponding to the NASA/National Bureau of Standards (NBS) Standard Reference Model for Telerobot Control System architecture (NASREM), which is the FTS system functional architecture [3]. It also defines software components to support communications between NASREM modules, task scheduling, and initial program load. This design consists of a set of top-level computer software components that correspond to processor and read-only memory device load modules and one set of lower level computer software components, most of which correspond to NASREM modules.

The detailed software design is expressed in the Ada Program Design Language (PDL), which is the adopted machine-compatible, higher order language for space station. The PDL permits the design to be expressed in such a way that a compiler can be used to check the consistency of interfaces. All Ada specification sections are provided in the detailed design, and components (e.g. functions, procedures) in the body sections are filled in to the extent that the compiler can perform its function. The remainder of the body may be filled in with Ada expressions or descriptions of the processing logic of the components.

Power Subsystem

The power subsystem interfaces electrically with the shuttle's power system, conditions this power to meet the needs of the telerobot and distributes the power to the DTF-1 subsystem loads. The power subsystem also performs the power switching required by subsystem loads; performs line filtering and EMI power quality filtering; provides power health status reports to the computers; and permits a safe power-up and power-down sequence.

The power subsystem generates three voltages from the 28 VDC the shuttle provides. These are 120 VDC for the motors and brakes, unregulated 28 VDC for the DMPS and thermal control subsystems, and regulated 28 VDC for the cameras and lights. Future missions will use a regulated and unregulated 120 VDC system.

Vision Subsystem

The vision subsystem includes one manipulator wrist-mounted camera, two head-mounted cameras, and camera lights. The cameras are color and use charge coupled device (CCD) technology. They and the lights are controlled from the workstation.

The vision subsystem will be used in conjunction with the shuttle's closed circuit TV system. The camera video output will interface to the shuttle's video switch and be displayed on the existing shuttle monitors. The output will also be recorded on the workstation video recorders.

The wrist camera gives the operator a view of the end effectors, permitting intricate tasks to be performed and allowing close-up inspections of completed work. The head cameras each provide fixed views of the worksite.

AFT FLIGHT DECK ELEMENT

The operator controls the telerobot and conducts the DTF-1 operations from the aft flight deck of the shuttle. The workstation is situated in the port-side corner of the aft flight deck close to the shuttle remote manipulator system (RMS) controls and uses the shuttle-provided TV monitors that are located in that corner.

The system is designed to be controlled by a single operator, although an observer may be used during the operation. Direct viewing of the telerobot will not be required; however the operator will be able to see it through the aft flight deck windows.

Workstation Subsystem

The workstation [figure 5] provides the man-machine interface in the shuttle aft flight deck for DTF-1 operations. It uses common hardware with the Space Station Freedom workstation and shuttle services hardware. The DTF-1 camera views will be displayed on the existing Shuttle displays. The shuttle-provided payload and

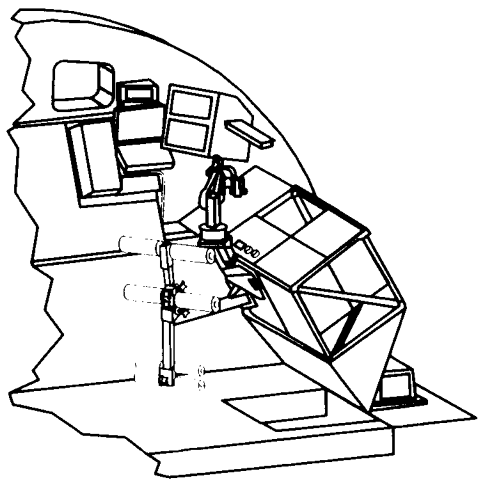


Figure 5. General Layout of the Aft Flight Deck for the Development Test Flight

general support computer (PGSC) will be used to display system status and function. The workstation operator uses the functions keys and the numeric and cursor keypad functions of the PGSC keyboard for menu and data entry operation. A separate control and display panel (C&DP) has a section of switches, indicators, and annunciators for manual control, emergency shutdown, and mode control. The PGSC and C&DP are mounted to a common plate, which is mounted to the A8 panel. The workstation configuration also incorporates a power control and distribution unit (PCDU), display assembly controller, a crew restraint system, a 6-DOF hand controller, and hand controller electronics.

Communication among the DTF-1 subsystem electronics elements is over a MIL-STD-1553b bus. This bus carries data from the workstation to the telerobot subsystems. Such data includes joint positions from the hand controllers and mode commands from the control and display panel. In the opposite direction, after being transformed by the telerobot control computer, manipulator force/ torque data is transmitted over the bus for feedback to the operator through the hand controller. Health and status and alert information is also transmitted over the bus for data storage, analysis, and display to the operator on the PGSC.

Within the workstation subsystem, the PCDU controls and monitors power distribution to the workstation electrical hardware. Video from the DTF-1 head and wrist cameras will be routed through the shuttle's video switch to the two shuttle monitors. These same signals will be recorded on the two shuttle video recorders located in the L10 panel. The operator will use the PGSC keyboard through the display assembly or the control and display panel controller to control the cameras.

Hand Controller

The DTF-1 hand controller is the Martin Marietta/Kraft 6-DOF, force-reflecting, hand controller [figure 6]. The hand controller is based on a mature design that has been used in nuclear and undersea applications since 1980.

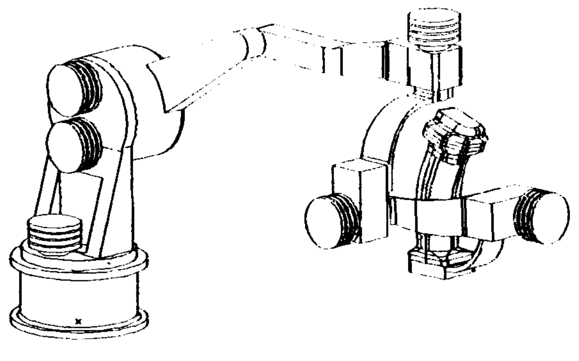


Figure 6. Hand Controller

The hand controller supports rate and position control with and without force reflection. It can provide 5 pounds of force and 9.5 inch-pounds of torque into the operator's hand. The hand controller electronics consist of a computer, analog to digital converter, input/output device, and pulse-width modulated power drivers.

Each hand controller joint consists of an induction motor and gearing, motor heat sink, potentiometer-based position sensor, and housings and bearings. The shoulder and elbow joints contain additional speed reduction. High gear ratios (200:1 in the wrist joints) are used in conjunction with low inertia motors and gears.

The detachable hand grip is similar to bottom-mounted, flight joysticks. It has an activation switch, which activates the hand controller and permits reindexing of the hand controller-to-manipulator transformation; an end effector enable switch, which enables the end effector to open or close; and the end effector rocker switch, which commands the end effector fingers to open or close.

Crew Restraint System

The crew restraint system consists of pairs of adjustable padded bars which act to restrain the operator from the hips down. It provides restraint in all axes, permitting safe force reflecting operations in zero gravity.

The restraint system is designed to accommodate operators sized from a 95 percentile American male to a 5 percentile Japanese female. The system also has a thigh crossbar vertical section that can rotate to place the pads against the front or rear of the thighs. The entire system can be rotated and tilted on the adapter so the operator can be placed at the centerline of the monitors and 28 inches away from the monitors. In this location, he/she can easily reach the C&DP and the PGSC and maneuver the hand controller throughout its operating envelope.

Hardwire Control

The C&DP has switches for bypassing the computers and directly controlling the telerobot, cameras and lights, individual manipulator joints, manipulator and mass caging mechanisms, and the end effector. This hardwire control system permits stowage of the payload in the event of certain failures of the DTF-1.

A safety emergency shutdown (ESD) switch on the C&DP allows the operator to safely shut down the telerobot. In the shuttle, this switch will be hardwired to the robot power distribution hardware. Activation of this switch will cut all power to the manipulator motors and brakes. The other subsystems will not be affected.

OPERATIONS

DTF-1 mission objectives will be met through a sequence of 12 tasks to be performed during the 16-hour mission timeline. The first is an initial system familiarization demonstration. The remainder of the mission activities are grouped into six performance verification tasks and five capabilities demonstration tasks.

SYSTEM FAMILIARIZATION DEMONSTRATION

This task will demonstrate that the DTF-1 system and the operator are ready for performing the on-orbit DTF-1 tasks. First, the manipulator and end effector will be positioned well away from any surfaces or objects that could be inadvertently contacted. Then the following tests will be conducted:

1) Test the boundary management/touch control system. The purpose will be to verify that motion outside the workspace will not occur. The operator will deliberately attempt to move the manipulator outside of the artificially established boundaries defined in the software specifically for this test.

2) Test the control available for single joints. This test will confirm if the operator can control the performance of single joints from the control panel.

Other tests will be to control and adjust the camera; operate the manipulator first with the handcontroller and then with the hardwire system; control the manipulator using various combinations of control mode, reference frames, and scale factors; control the gripper with the C&DP and hardwire system; and reindex the hand controller.

PERFORMANCE VERIFICATION TASKS

Fine Positioning Test

This test is an automated sequence that will test the performance of the manipulator to ensure that accuracy, incremental control, and repeatability performance requirements are met. The accuracy test will be performed using the wrist camera and an inverse prospective technique. The repeatability and incremental motion tests will be performed using joint position sensors and forward transformation to the tool plate. The ISO definitions, equations, and approaches will be used to determine the results.

Operational Envelope Evaluation

This automated sequence will evaluate the performance of the manipulator at workspace extremes, test enroute velocity and workspace limits, and demonstrate recovery when limits are exceeded.

Manipulator Dynamics Model Verification

This task has two purposes: Test a joint closed loop actuator and structural dynamics model and verify on-orbit stability margins and the performance of the position-based impedance control.

For the joint closed loop actuator and structural dynamics model test, automated sequence inputs will be given to the seven joints, one at a time. The test will be performed for three different arm configurations, with the arm unloaded and then loaded with the gripper holding the 25-pound mass. The mass will be grasped to verify the performance with different inertias.

For the automated impedance test, the manipulator will be rigidly connected to the center handle of the caged mass and then a step force or position command will be given in all 6-DOF.

For the teleoperated tests, the operator will use the handcontroller and different force reflection gains to command gross motions of the loaded manipulator. Different spring return forces will be used to test the resolved rate mode and determine its operation.

Manipulator Non-linear Model Verification

This task provides the data to characterize and verify the non-linear model of the joints. Automated signal input sequences will be input, one at a time, to shoulder pitch, elbow pitch, and wrist pitch. Joint brakes except for the joint being simulated will be ON. This task will be performed for three thermal conditions and in conjunction with the thermal transient response tests.

Thermal Transient Response

This task will test the thermal capacitance of the thermal control subsystem while evaluating manipulator performance under different thermal environments.

CAPABILITIES DEMONSTRATION TASKS

Peg-in-Hole Demonstration

An operator will use teleoperator control to insert a peg mounted on one of the fingers into four different-sized holes. A video of the insertion and a transcript of the operator's comments will provide data, including preciseness of peg insertion, depth of insertion, difficulties completing the task, and human-machine interface data, such as control preferences and hand controller feel, predicted stability, operator fatigue, and teleoperational mode performance margins.

Contour Board Tracking

This task will evaluate impedance control and force reflection during end-point tracking tasks and provide engineering data for evaluating the human-machine interface. An operator will use teleoperation to control a peg in tracking curved, straight-line and V-shaped machined tracing paths, 3-D trajectories over rough surfaces, sloping plane surfaces and convex, or concave contour surfaces. The task will be performed with and without force reflection but with active impedance control. Data collected will include the operator's comments during the task and a video of the peg tracing the surface.

Connector Demate/Mate

The gripper will demate and mate a Symmetrics connector, which will require multiple revolutions of its collar for locking and unlocking. This task is representative of an FTS task to be performed on Space Station Freedom. The engineering data generated will be used to determine the operational performance of the impedance controller, bilateral force reflection, and manipulator safety limits and to evaluate the human-machine interface. The gripper will unlock and separate the connector halves. Then the gripper will mate the connector halves, lock them, and apply a small lateral force to verify the integrity of the connection. Throughout the repetitions of this task, the force reflection gain will be varied as a function of different gains. Task completion times and contact force levels will be used to measure operator fatigue.

Truss Strut Element Removal/Replacement

This task is another FTS-like task in which a Space Station Freedom truss strut element will be removed and replaced. The gripper will grasp the collar on the truss strut element mounted on the task panel and unlock and relock it. The strut will then be partially demated from the strut attachment fitting mounted on the node. Engineering performance data on the impedance controller, force reflection, and manipulator safety limits will be generated. Human-machine interface and operator fatigue will be evaluated from operator comments and a video of the task.

Removable Mass Manipulator Loading Test

A 25-pound removable mass that is caged to the task panel is provided to enable evaluation of the manipulator performance in both a loaded and unloaded configuration. Two separate handles are provided on the mass for gripper attachment. The handle designs will incorporate mating interfaces matching the dedicated handle grasping interface of the end effector. One handle is located on the center of gravity of the mass and the other is located off the center of gravity in order to evaluate different inertial loadings of the manipulator. The removable mass is the only removable part of the task elements. A caging mechanism, similar in design to the lower arm caging mechanism, is provided for the restraint of the mass.

SAFETY

The design of the DTF-1 system is driven heavily by safety considerations. Mission procedures focus on maintaining the integrity of the shuttle and protecting the crew. The DTF-1 design ensures survivability after being subjected to normal or emergency landing loads and post-landing delays.

The telerobot may be shut down in three modes: Normal shut down, manual emergency shut down, and automatic emergency shut down. In normal shut down, the operator can command the telerobot to shut down after it has completed work or when a failure occurs that is not hazardous to the crew. The shutdown command may include stowage of task element hardware.

In manual emergency shut down, the operator may command shut down of the telerobot from the DTF-1 workstation at which DTF-1 operations are being monitored and controlled. This mode will be used to respond to DTF-1 failures that might be hazardous to the crew, the shuttle structure, or the telerobot.

Automatic emergency shut down will allow the telerobot to shut itself down in the event of a self-diagnosed failure or other unsafe condition. Such conditions include the failure of the shuttle or support system to supply required power and/or data links and the possibility of the telerobot colliding with itself or other structures. The telerobot will shut itself down automatically when these failures occur or when an out-of-limits condition exists. After corrective actions or workarounds have been implemented, the operator can go through the normal startup and checkout procedures to continue working. If the arm becomes partially inoperable, degraded mission operations could continue. The safety of the crew and the shuttle will be considered to be paramount. Those tasks that require full operability and that were not completed when the arm failed will be removed from remaining mission profile timeline. The hardwire control scheme will be implemented to restore the manipulator in the event a failure precludes computer control of the arm.

In the unlikely event that a failure or series of failures prevents the manipulator or removable mass from being restowed by the normal method or by hardwire control, there is equipment that will allow the DTF-1 payload to be jettisoned using the shuttle RMS.

A third hazard control will be to use EVA to restow the payload for safe shuttle reentry and landing. The manipulator and gripper are designed so that an astronaut on EVA can manually release the brakes on them if they fail and backdrive the system into the correct position for caging. If a manipulator joint seizes, thereby preventing restow, the astronauts can remove the manipulator at the shoulder and stow it on the failed manipulator arm storage system (FMASS) which is attached to the MPESS.

CONCLUSION

During the 16 hours of mission operation, the Development Test Flight will return valuable engineering data on the performance of the FTS manipulator in zero gravity and on the human-machine interfaces necessary for the efficient operation of a teleoperated system from the aft flight deck of the shuttle. This data will be analyzed post flight, and the results will be used in the development of the final flight system which will be used in the assembly and maintenance of Space Station Freedom.

REFERENCES

1. Andary, J.; S. Hinkal; and J.Watzin; "Design Concept for the Flight Telerobotic Servicer (FTS)," presented at the 2nd Space Operations Automation and Robotics Workshop (SOAR88), Dayton, OH, July 20-23, 1988; NASA Conference Publication CP-3019, pp. 391-396.

2. Andary, J.; K. Halterman; K. Hewitt; and P. Sabelhaus; "The Flight Telerobotic Servicer (FTS): NASA's First Operational Robotic System," presented at the 3rd Operations Automation and Robotics Workshop (SOAR89), Houston, TX, July 25-27, 1989; NASA Conference Publication CP-3059, pp. 311-318.

3. Albus, J.; H. McCain; and R. Lumia; NASA/National Bureau of Standards (NBS) Standard Reference Model for Telerobot Control System Architecture (NASREM), SS-GSFC-0027, December 4, 1986.

N91-20662

DEPLOYABLE ROBOTIC WOVEN WIRE STRUCTURES AND JOINTS FOR SPACE APPLICATIONS

Mo Shahinpoor
Department of Mechanical Engineering
University of New Mexico
Albuquerque, New Mexico 87131

Bradford Smith
Woven Wire Corporation
Santa Fe, New Mexico 87502

ABSTRACT :

Deployable robotic structures are basically expandable and contractable structures that may be transported or launched to space in a compact form. These structures may then be intelligently deployed by suitable actuators. The deployment may also be done by means of either air–bag or spring–loaded type mechanisms. The actuators may be pneumatic, hydraulic, ball–screw type or electromagnetic. The means to trigger actuation may be on–board EPROMS, programmable logic controllers (PLC's) that trigger actuation based on some input caused by the placement of the structure in the space environment. The actuation may also be performed remotely by suitable remote triggering devices. In this presentation several deployable woven wire structures are examined. These woven wire structures possess a unique form of joint, the woven wire joint, which is capable of moving and changing its position and orientation with respect to the structure itself. Due to the highly dynamic and articulate nature of these joints the 3–D structures built using them are uniquely and highly expandable, deployable and dynamic. They naturally give rise to a new generation of deployable three–dimensional spatial structures.

INTRODUCTION:

Woven wire structures are comprised of intersecting multi–wire elements which are preferably capped and possess sliding positionable retainers. The unique feature of these structures is a joint called the woven wire joint (Patent Pending) which is capable of shifting its position with respect to the structure itself in a three–dimensional manner. Traditionally a lower pair prismatic joint is the only classical joint capable of shifting its position with respect to its structure in a linear fashion. The classical joints fall into two categories of

1– Lower–pair joints (LPJ)
(surface–to–surface contact)

which comprise, revolute joints, (Fig. 1a) prismatic joints, (Fig. 1b) sliding joints (Fig. 1c), and ball–and –socket or spherical joints (Fig. 1d).

2– Upper–pair joints (UPJ)
(point–to–point or line–to–line contact)

which comprise a variety of contact joints as shown in Figures (2a,2b,2c,2d).
In the context of the above two categories the woven–wire joints fall in a separate category which may be appropriately called internal or intermeshing–pair joints, or IPJ's as shown in Figure 3. Thus, here is introduced a new category of joints, namely

3– Internal or Intermeshing–Pair Joint (IPJ)
(a collection of line–to–line contacts)

These woven wire joints are comprised of intersecting multi–wire elements as shown in Figures 3 and 4. Similar to other classical joints, there exist a variety of topological configurations for these woven wire

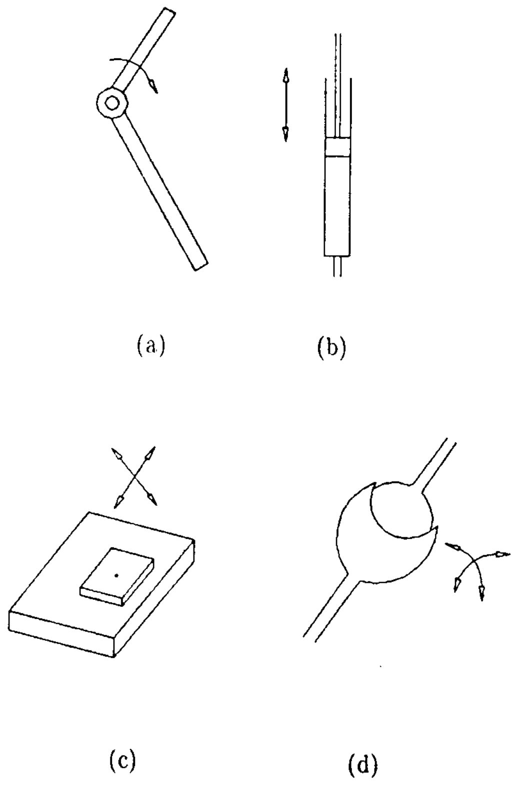


Fig. 1– Lower–Pair Joints

joints which will be discussed in this article. However, before elaborating on various ramifications of woven–wire structures it will be appropriate to briefly discuss the general nature of spatial structures in the context of Peter Pearce's minimum inventory/maximum diversity principle as well as Wiliam P. Thurston's generalization of three–dimensional manifolds.

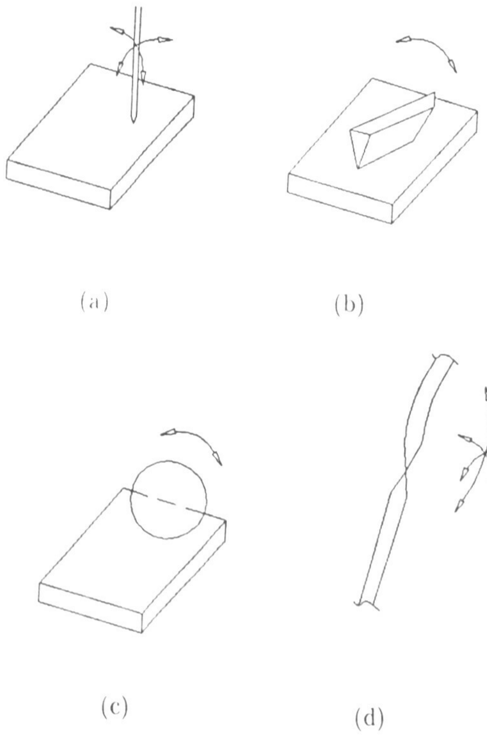


Fig. 2– Upper–Pair Joints

Geometry and geometrical configurations are the fundamental disciplines based on which the entire universe is constructed. The mathematical topology of three–dimensional manifolds reveals that most of the manifolds can be analyzed by means of geometry. The construction process is the motion of primitive entities. The most primitive entity is a point whose translation constructs a line. The translation and rotation of a line in multitude, is capable of generating all possible geometrical entities in the entire universe.

The study of three–dimensional manifolds is, thus, a generalization of two–dimensional manifolds which, itself, is a generalization of one–dimensional manifolds.

Due to the complex structure of some three–dimensional forms no complete classification of three–manifolds has, thus far, been achieved. Based on work by William P. Thurston at Princeton University it is possible to generate all possible three–manifolds by means of a specific pattern. This implies that all twistings and windings of three–manifolds can be described in geometric terms. Peter Pearce's book "Structure in Nature is a Strategy for Design", MIT Press 1980, has presented a systematic and to some extent comprehensive study of 3–D spatial systems. Clearly, Circles can give rise to cubes and polyhedra.

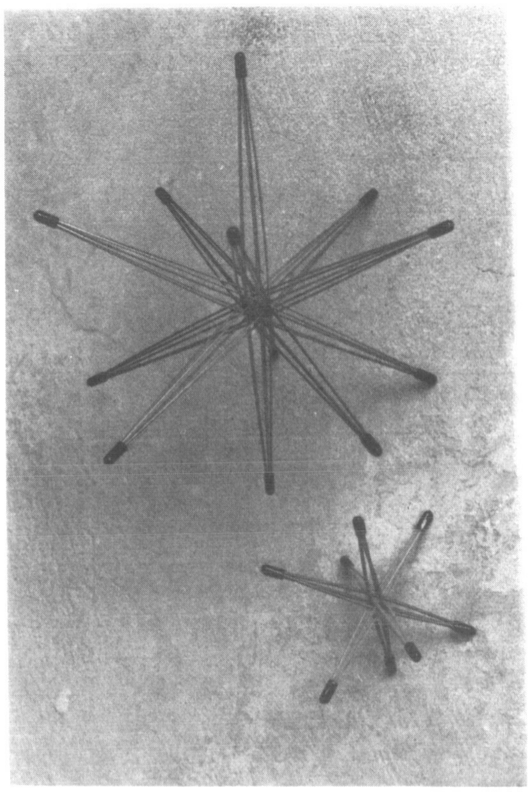


Fig.3– Two different configurations of woven wire joints

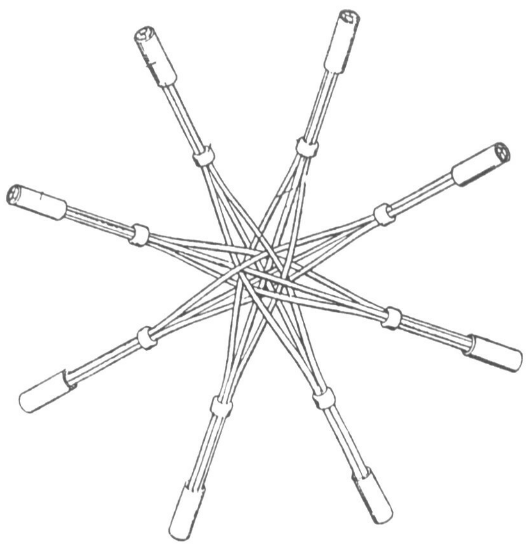


Fig. 4– The three wire, six element (36) weave

Polyhedra can, in turn, generate all possible geometrical entities. As originally suggested by Herbert Seifert and C. Weber in 1933, the three–manifolds can be understood from a polyhedron by abstractly glueing certain faces together. By abstractively glueing the opposite Pentagons of a dodecahedral the Seifert–Weber dodecahedral space can be constructed.

Figures 5 and 6 illustrates some fundamental relationships between motions of a point in three–dimensional space and the evolution of Platonic solids, i.e., tetrahedron, cube octahedron, and dodecahedron.

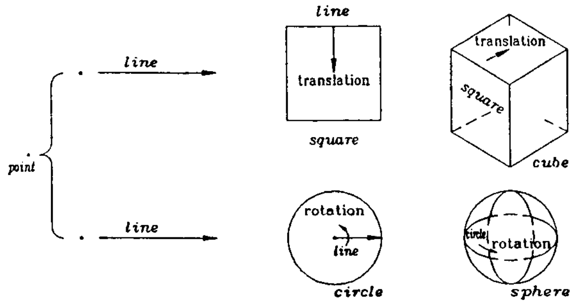


Fig. 5– Evolution of Primitive Solids

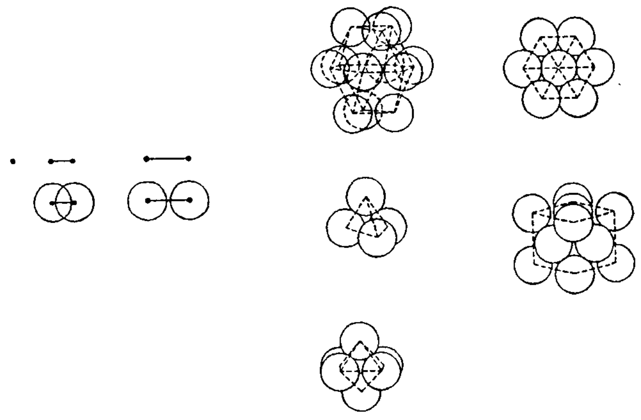


Fig. 6– Evolution of Platonic Solids From Spheres

A useful metaphor to study the complex nature of three–manifolds is the use of mechanical systems of linkages. A mechanical linkage is represented mathematically by a set of line segments in the plane with pivot points at line intersections. Mathematically, it is assumed that the lines and pivot points can pass freely through one another. Woven–wire joints and structures approximate such an idealization with a great degree of accuracy. Up until now, constructing a physical model whose bars and joints replicate the idealized linkage has not been possible. For any mathematical version of a linkage system there exists a physical linkage that produces the same motion. However, the actual physical linkage may be quite different, geometrically, from the ideal linkage. The "configuration space" of a linkage system is the set of all of its possible positions in the three–dimentsional space.

In order to understand the configuration space of linkages one should consider the possible configurations of the double–linkage mechanism regardless of position of its free end (Fig. 7).

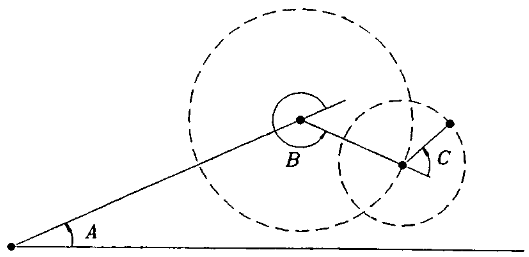


Fig. 7– Double and Triple Linkage Configuration Space

Every configuration of the double–link system can be described by two angles A and B. If a third linkage is considered, a third angle C is needed to completely specify the associated configuration space. Every possible position of the triple crank is represented by a unique point in a cube whose apposite pairs of faces are abstractly glued together.

The work space of a triple–double cranks joined by one end pin (Fig. 8a) is an 8–dimensional space in which every point corresponds to 8 different configuration of the system (Fig. 8b). In this space the boundaries are four–dimensional spaces (Fig. 8c) and the vertices are two–dimensional surfaces (Fig. 8d).

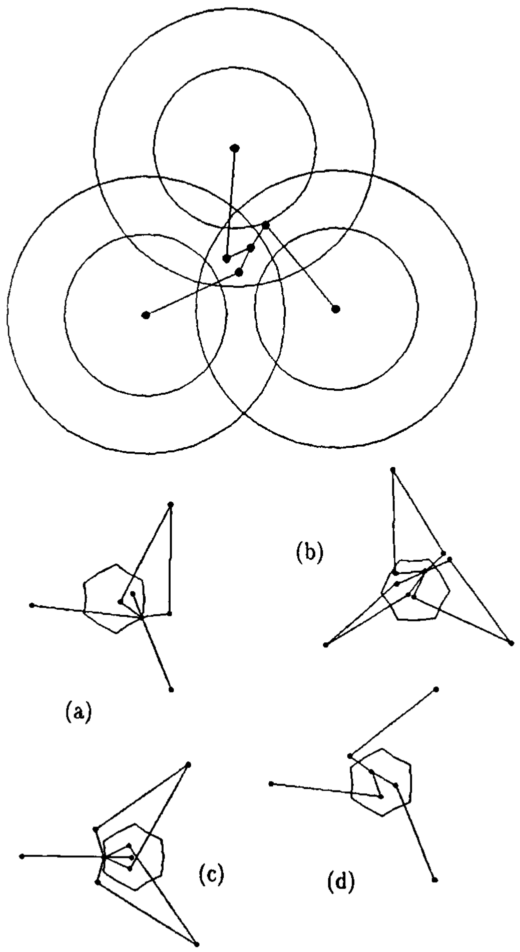


Fig. 8– The Work Space of a System of Three Double–Cranks

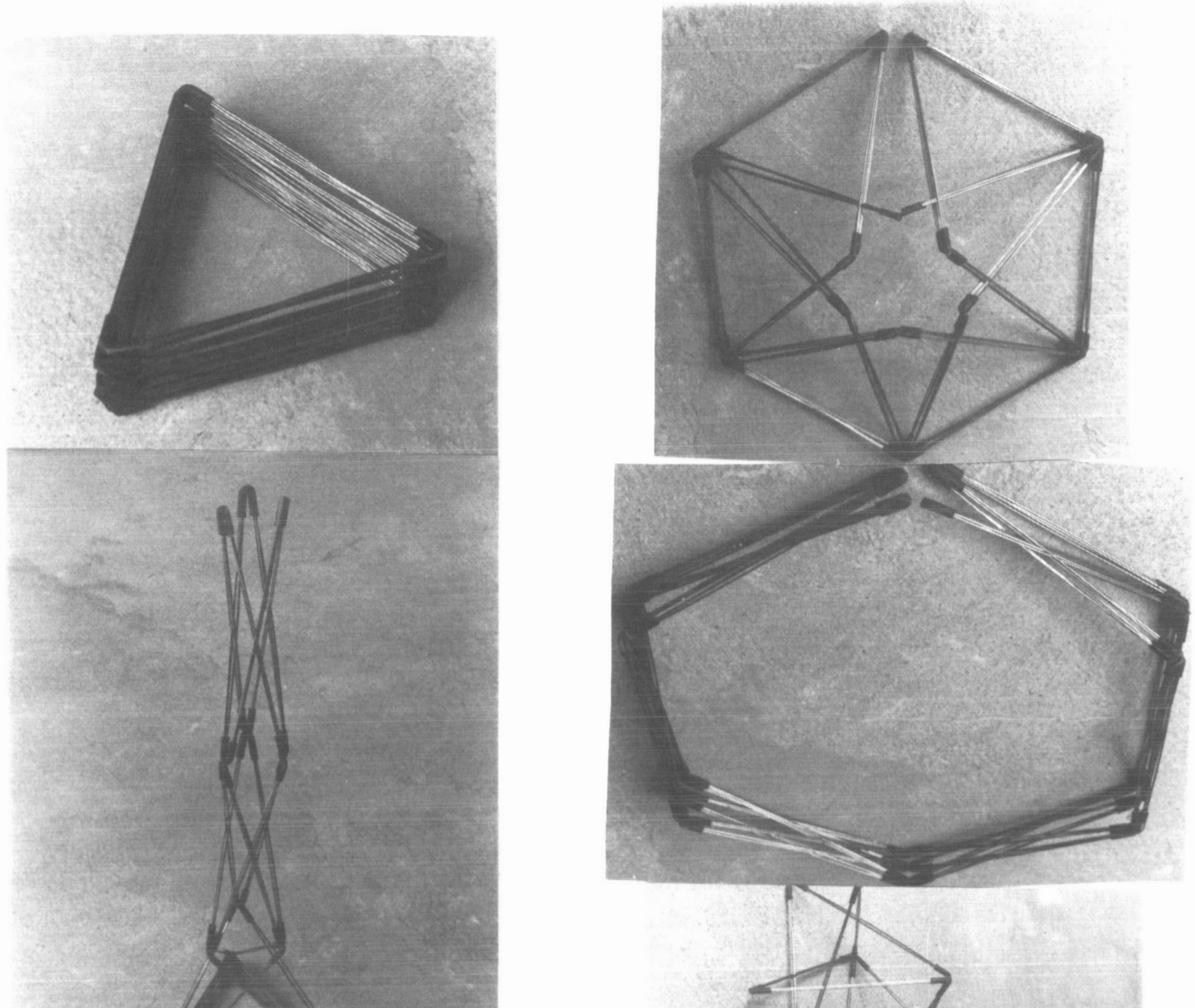


Fig.9–A Variety of Woven Wire Configurations

The woven–wire structures, in the context of linkages, define highly complex higher dimensional work spaces. The woven–wire structures and joint can also be considered in the context of Peter Pearce's systems of diversity; minimum inventory/maximum diversity systems. Endless variety of three–demensional structures can be constructed by a set of one, two, or three woven–wire joints (Fig. 9a,b,c,d,e,f). A fundamental property of minimum inventory/maximum diversity systems is the principle of conservation of resources. This principle sometimes manifests itself as the principle of minimum potential energy for greater stability. For example the closest packing of entitities is a structural arrangement of inherent geometric stability creating three–dimensional arrangements of polyhedral cells in living and non–living systems. If, for instance, the centers of closest packed equal spheres are joined, a three–dimensional arrangement of equilateral triangles is created. It is well known that triangulated frame–works exhibit inherent geometric stability because they are stress–compensated. Woven–wire structures are also inherently triangulated and polyhedral in nature.

Figure 10 shows a collapsible woven–wire structure that can be accordioned by a user, either horizontally or vertically. The wire bundles are movable in concert about their woven–wire (weave) joint.

A woven wire apparatus may be comprised of at least one bundle, which bundle comprises a plurality of multi–wire elements. Each multi–wire element has a first end and second end, and each multi–wire element further comprises a plurality of stiff but slightly flexible wires of substantially similar length. Each multi–wire element comprises a structure for positioning the multi–wire elements in a mutually intersecting relationship to one another. The apparatus preferably has a structure for joining the first end and a structure for joining the second end of the multi–wire elements whereby the elements are retained in intersecting relationship to one another. The wire retaining structure fixes the elements in position. The end joining structures are removably positioned on the ends of the multi–wire elements and comprise generally tubular caps, flat bases, suction bases, balls, padded bases, or the like. The end joining structure of the apparatus preferably comprises rigid connectors, movable connectors, or a mechanical hinge structure, but may further comprise color coding.

A woven–wire structure further comprises wire retaining structures which are generally cylindrical and slidably and romovably positionable on multi–wire elements. The wire retaining structure preferably comprises color coding and a ring, tube, strap, clip, or the like, fixable about the joint.

These structures comprise mutually intersecting multi–wire elements which further comprise a like plurality of wires. The multi–wire elements may comprise wires of substantially similar length or substantially different length. The bundles in a woven–wire structure further comprise support members.

The multi–wire bundle of a woven–wire structure is collapsible and comprises an odd number of wires, preferably at least three wires. Multi–wire elements may be chosen with the same length to provide a generally symmetric apparatus.

The woven–wire structures are useful for holding an object within the joint. They may further comprise two or more bundles positionable in an adjoining relationship being stackable or positionable side by side or stackable. The bundles preferably comprise stabilizing means. Bundles of the preferred embodiment comprise stabilizing structures. They are useful as a support structure or a cover, when the bundles are positionable in a substantially concave shape. The bundles are also positionable in a spherical shape.

A woven–wire apparatus with at least one bundle, comprising the additional structures for joining the first end and second ends of the multi–wire elements, whereby the elements are thereby retained in said intersecting relationship to one another, further comprising dynamic means also provides alternating motion of the multi–wire elements. The bundle is fixed in position after motion is provided, by dynamic means. A woven–wire joint creates a spherical and structurally sound joint using intersecting wire bundles. Such joints are usable in a wide variety of structures. The movability and generality of such joints makes them far superior to traditional lower or upper pair joints.

The accompanying drawings and photographs (Figures 11 through 20) illustrate several embodiments of the woven–wire structures and, together with the description, serve to explain the potential and the beauty of woven–wire structures and joints for space applications.

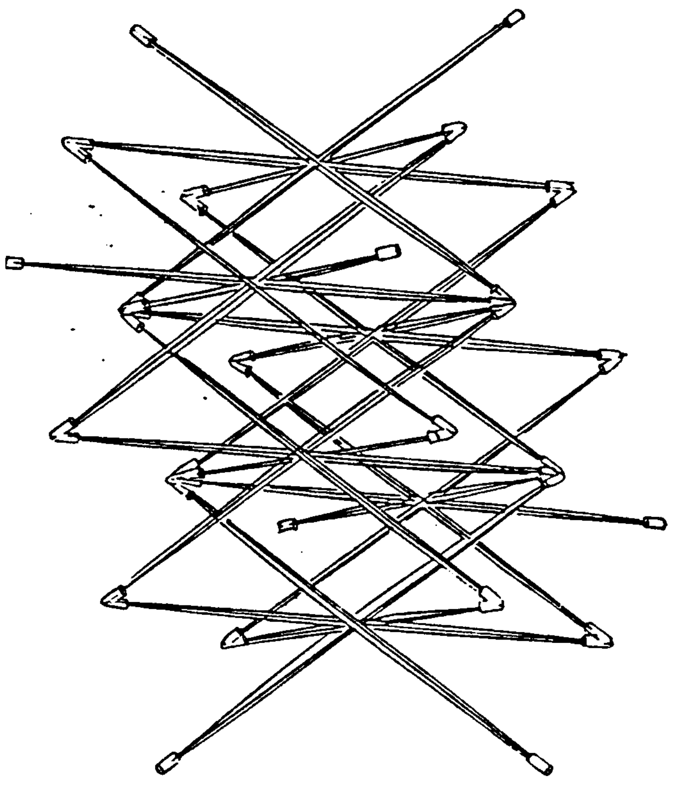
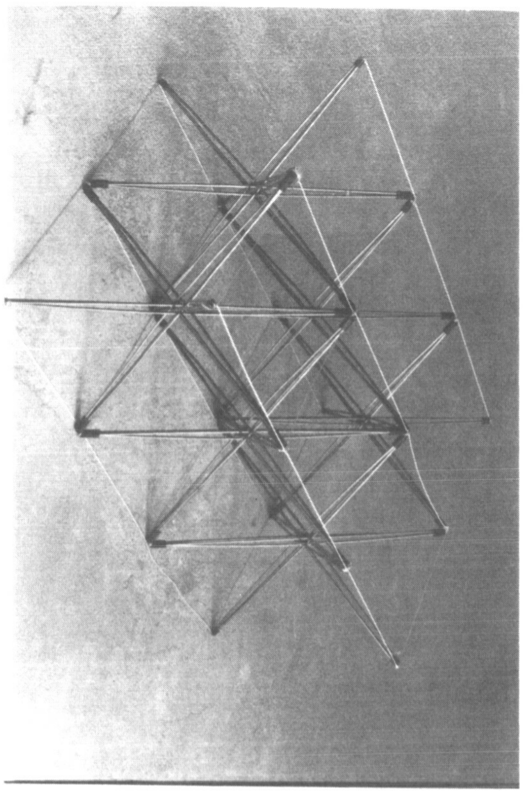


Fig. 10– A collapsible Woven Wire Structure



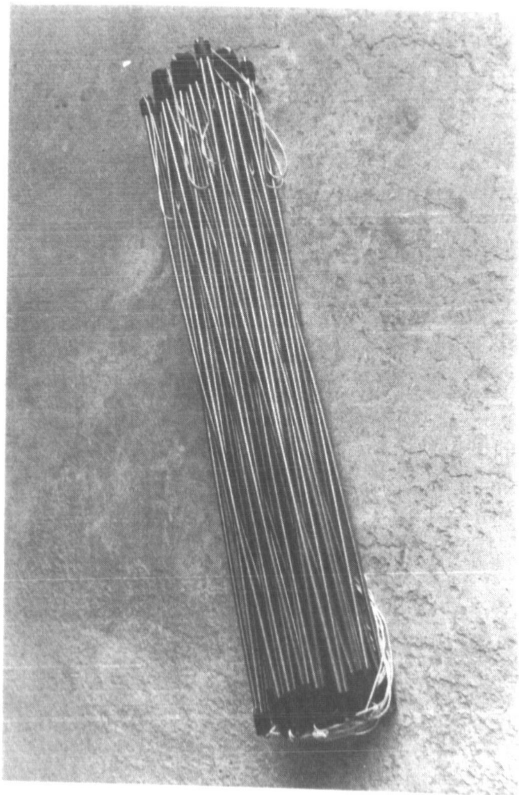


Fig. 11– A collapsible woven wire net of 6 individual modules.

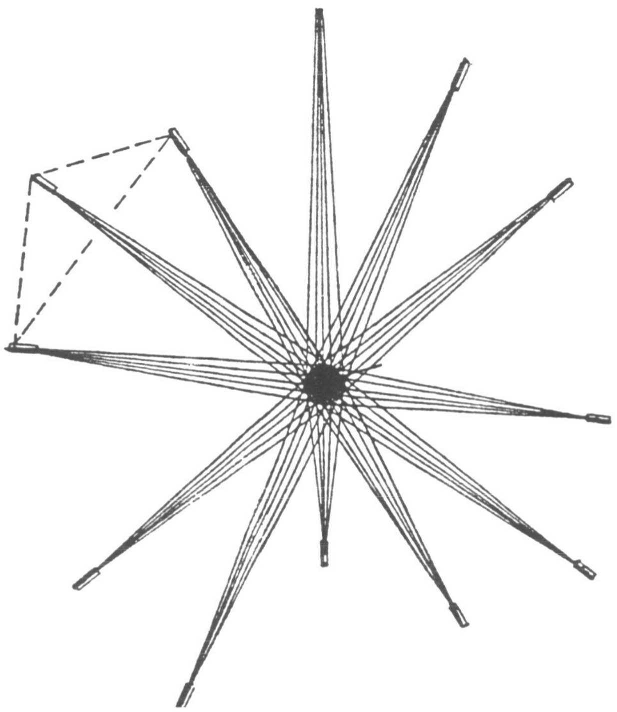


Fig. 12– A woven wire joint holding a sphere.

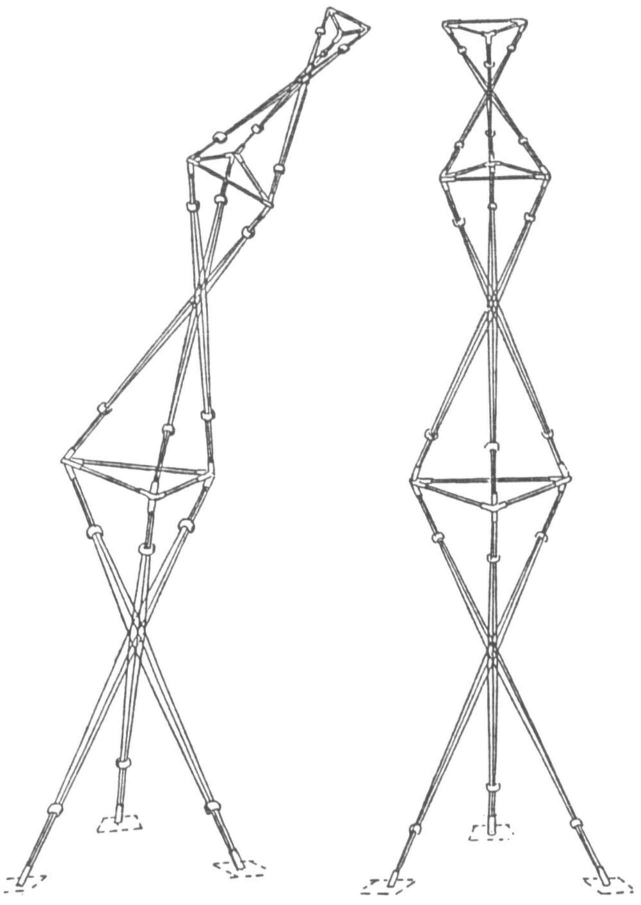


Fig. 13– The tower–type structure made from woven wire joints

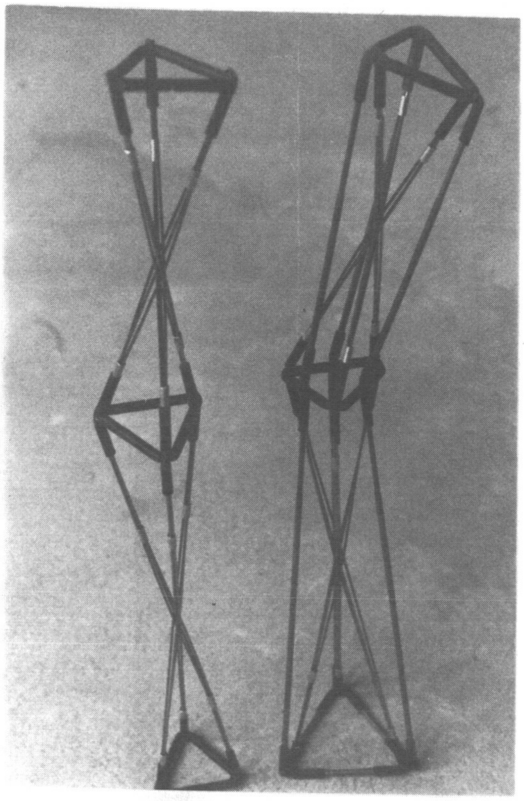
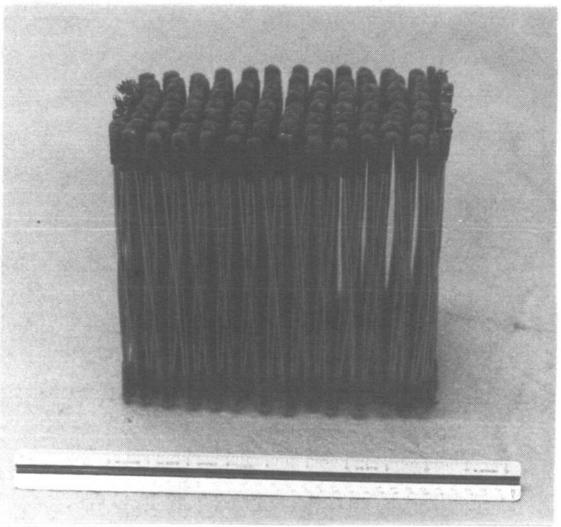
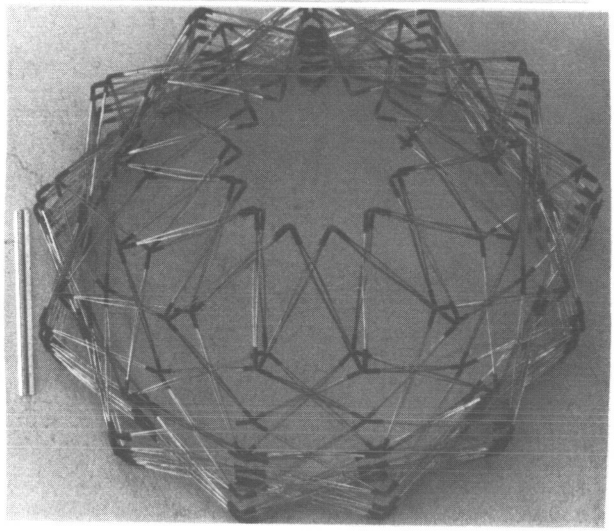
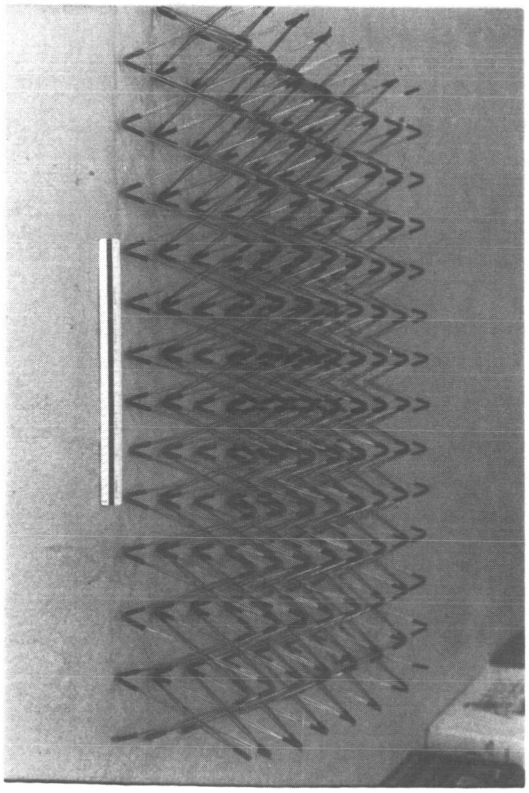


Fig. 14– Constructed tower–like woven wire structures







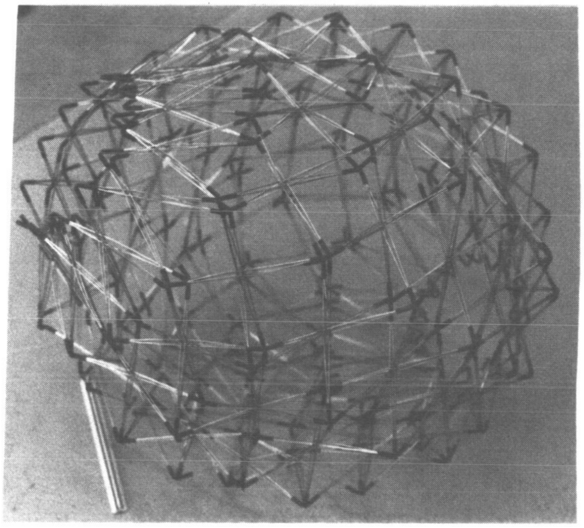


Fig. 15– Various configurations of woven wire dome structures

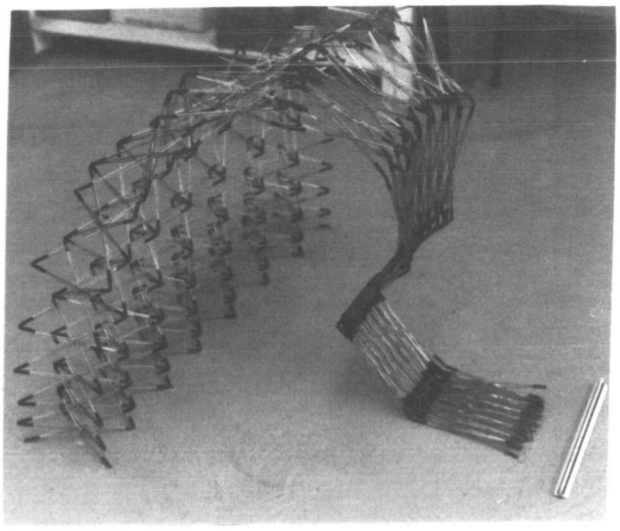


Fig. 16– Woven wire net strucutres

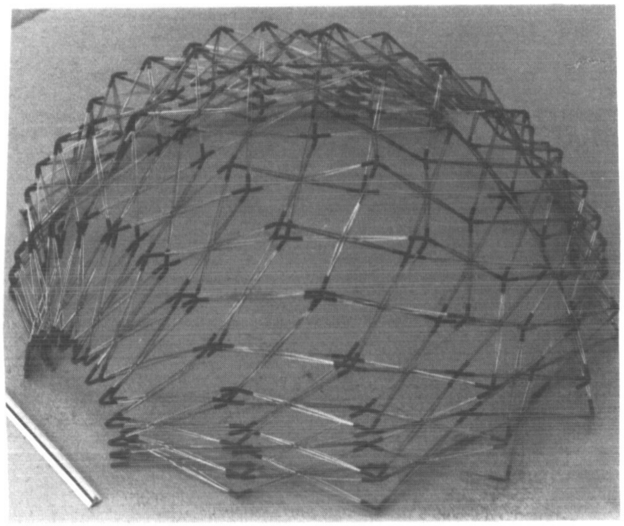


Fig.17– Various woven wire dome structures

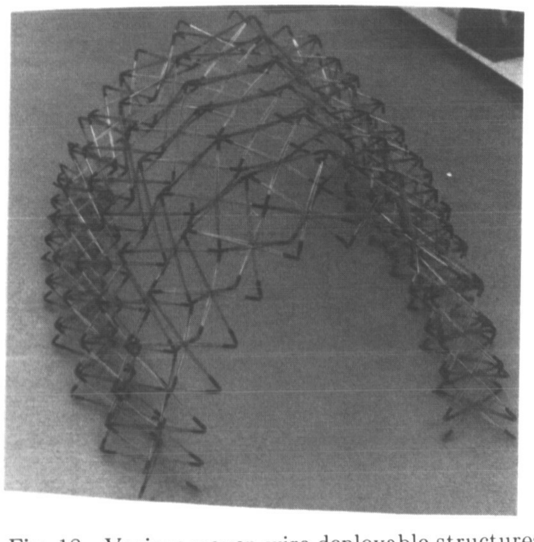


Fig. 18– Various woven wire deployable structures

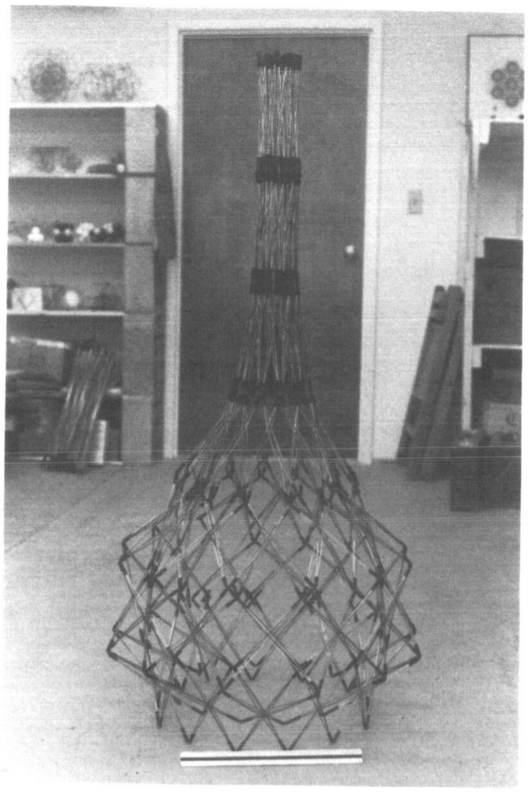


Fig. 19– Various deployable bottle–like structures

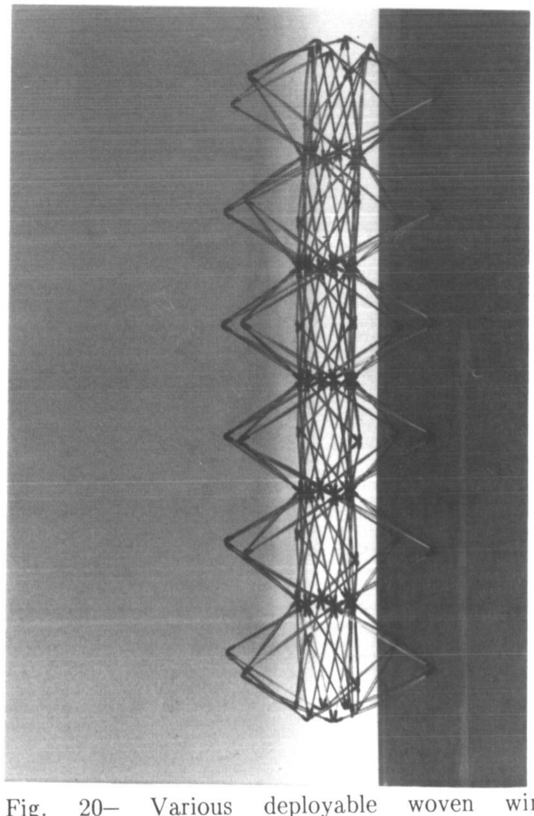


Fig. 20– Various deployable woven wire tunnel–like structures

ROBOTIC APPLICATIONS WITHIN THE STRATEGIC DEFENSE SYSTEM

Dale A. Nussman
Dynamics Research Corporation

(Paper not provided by publication date.)

TELEROBOTICS

PRECEDING PAGE BLANK NOT FILMED

TELEROBOTIC PERFORMANCE ANALYSIS

Neil A. Duffie
University of Wisconsin

(Paper not provided by publication date.)

TELE-AUTONOMOUS CONTROL - THEORY AND PRACTICE

Ilhan Ince, John Garin, and Keith Bryant
Martin-Marietta/Baltimore, MD

(Paper not provided by publication date.)

N91-20663

INVESTIGATION OF VARYING GRAY SCALE LEVELS FOR REMOTE MANIPULATION

John M. Bierschwale, Mark A. Stuart, and Carlos E. Sampaio
Lockheed Engineering and Sciences Company
Houston, TX 77058

ABSTRACT

A study was conducted to investigate the effects of variant monitor gray scale levels and workplace illumination levels on operators' ability to discriminate between different colors on a monochrome monitor. It was determined that 8-gray-scale viewing resulted in significantly worse discrimination performance compared to 16- and 32-gray-scale viewing and that there was only a negligible difference found between 16 and 32 shades of gray. Therefore, it is recommended that monitors used while performing remote manipulation tasks have 16 or above shades of gray since this evaluation has found levels lower than this to be unacceptable for a color discrimination task. There was no significant performance difference found between a high and a low workplace illumination condition. Further analysis was conducted to determine which specific combinations of colors used in this study can be used in conjunction with each other to ensure error-free color coding/brightness discrimination performance while viewing a monochrome monitor. It was found that 92 three-color combinations and 9 four-color combinations could be used with 100% accuracy. The results can help to determine which gray scale levels should be provided on monochrome monitors as well as which colors to use to ensure the maximal performance of remotely-viewed color discrimination/coding tasks.

INTRODUCTION

Telerobotic workstations will play a major role in the assembly of Space Station *Freedom* and later in construction and maintenance in space. For the short-term, control of these telerobotic systems will be dependent primarily on the human operator. Since the human operator will be a part of the telerobotic system, it is critical that the components of this interface be designed so that the human operator's capabilities and limitations are best accommodated within the structure of specific task requirements. To emphasize the importance of a well-designed human-telerobot interface, one study found that the selection of an appropriate control device, based upon the operator's capabilities and the requirements of the task, can more than double the productivity of the telerobotic system (O'Hara, 1986).

One of the most important components of the workstation will be the vision system. During many tasks, where a direct view is not possible, cameras will be the user's only form of visual feedback. Monochrome viewing has been used in the past for remote manipulation and inspection. Even though color operations have been baselined for *Freedom*, special situations may dictate the use of monochrome viewing. Other telerobotics users such as future space systems, space station investigators, the nuclear industry, and undersea industry may find the use of monochrome viewing to be advantageous.

An important system parameter for monochrome viewing is number of gray scales that can be displayed on a monitor. The term "gray scale" refers to the uniform variation from black to white through various shades of gray in a television screen image when cathode ray tube (CRT) control voltages are adjusted over the full range of brightness for a specific monitor. The number of gray scales is an issue because more gray scale divisions require greater band width and processing capabilities which are often limited in remote environments. Therefore, the optimal number of gray scales from an economic or cost effectiveness standpoint will be the smallest number of gray scales that provide acceptable performance by the operator. 8, 16, and 64 gray scale values are being considered by engineers for displays.

A survey of the literature has failed to provide definitive guidelines concerning this issue (e.g., Johnston, 1968; Troy, Deutsch, and Rosenfeld, 1973; Shurtleff, 1980; and Kingdom and Moulden, 1986). Woodson (1980) recommended that a minimum of five gray levels be used for monitors, but precise quantification of the specific gray scale levels to use was not found. Tannas (1985) stated that contouring in big, tone-changing areas is bad if the number of gray scales is less than 16 and that 64 shades of gray should be used for good, aesthetic picture quality images. Tannas also stated that good gray scale performance requires a pixel contrast ratio of approximately 20:1 with the luminance either continuously variable or controllable into at least 16 logarithmically spaced steps. There is little research on the effect of variant gray scales on the performance of an operator performing a remotely-viewed task.

The computer equipment used in this evaluation had the capability of displaying between 2 and 256 different gray scales. Since it was impossible to compare all levels, the first objective of this study was to determine which of the possible gray scale levels are discriminable from each other.

One telerobotic task which would be performed on *Freedom* would be the assembly or maintenance of a thermal utility connector. Color coding may be used to indicate when the valves on each hose are either fully opened or closed. This would be helpful when the task is viewed on a color monitor or performed by an extravehicular astronaut. If this task were viewed on a monochrome monitor, then the information that is used to discriminate between the two colors is the brightness level of the colors. The second objective of this study was to study the discrimination performance of operators while they viewed colored visual stimuli through a monochrome monitor using the pre-determined number of variant gray scale conditions.

Since lighting conditions vary greatly in outer space, the third objective of this study was to also investigate the effect of different illumination levels on the experimental task. The performance of the color discrimination task took place under two different illumination levels as well as variant gray scales. (For a discussion on the effects of lighting on remote manipulation tasks see Chandlee, Smith, and Wheelwright, 1988).

These three objectives were investigated by conducting two different evaluations. The first evaluation narrowed down the number of gray scales while the second evaluation addressed the remaining two objectives.

EVALUATION 1

The objective of Evaluation 1 consisted of determining the discriminable gray scale levels to be investigated in Evaluation 2.

Subjects

Seven volunteer male subjects were selected to participate in this evaluation. All subjects who participated had their vision tested at the JSC Clinic and it was determined that they all had either corrected or uncorrected 20/20 vision and none had evidence of color deficiencies.

Apparatus

Testing took place in the Remote Operator Interaction Laboratory (ROIL) of the NASA Johnson Space Center. The video signal from a digital camera (focused on an extra-vehicular activity (EVA) toolbox) was processed through a DataCube digital image processing system and displayed on a Conrac monitor. The DataCube's output was a monochrome image with one of the following gray scale levels: 8, 16, 32, 64, 128, and 256. The illumination level in Evaluation 1 was the laboratory ambient lighting level of 269 lx (25 fc).

Procedure

The procedure followed was a paired-comparison psychophysics technique. Each operator was randomly presented the image using one of the six gray scale levels as a reference. The reference was then successively paired to each of the other gray scale levels so that a comparison could be made. The reference gray scale was also paired to itself as a control condition. The pairs were formed serially in this part of the testing. Immediately after each subject viewed each of the reference-comparison pairs, they would then state whether or not there was a perceptible difference between the two stimuli. After completing these six paired-comparisons, another of the remaining five gray scale levels was selected as the next reference. Six more paired-comparisons were then conducted. The procedure continued in this fashion until all six of the gray scale levels had served as the reference gray scale level for this paired-comparison task.

Results and discussion

The data were analyzed in terms of determining the number of subjects who noticed a difference between a specific gray scale level and the other gray scale levels during the paired-comparisons tasks. If a difference was not noticed, then this was classified as an error.

The discrimination error-rates were then statistically analyzed with an analysis of variance (ANOVA) procedure. Based upon the results of the data analysis, it was determined that there was a significant effect ($p < 0.05$) due to the different gray scale levels used. A Newman-Keuls pairwise comparison statistical procedure was then administered to these data. The Newman-Keuls revealed a consistent trend for subjects to notice a difference between 8 shades of gray and all other levels and between 16 shades of gray and all other levels. This trend was not observed for discriminations with 32, 64, 128, and 256 shades of gray.

Based upon the results of the ANOVA and Newman-Keuls analyses, it was concluded that subjects were not able to discriminate between 32, 64, 128, and 256 shades of gray for static monochrome viewing. Because of this conclusion, it was decided to evaluate only one gray scale level from this group during the second evaluation of this study. Thirty-two shades of gray was selected for study, although any of the other three levels could have just as well been selected. Since subjects were consistently able to discriminate between 8 and 16 shades of gray and all the other levels, then these two gray scale levels were also included in this study's second evaluation. Therefore, Evaluation 2 studied the effects of 8, 16, and 32 shades of gray on operator perceptual discrimination performance.

EVALUATION 2

The objective of Evaluation 2 was to determine how color discrimination performance is affected by variant gray scale and worksite illumination conditions while viewing a monochrome monitor.

Subjects

Twelve volunteer subjects were selected to participate in this evaluation. Seven subjects were male and five were female. Subjects were randomly assigned to one of two different groups. These two different groups represented the different illumination conditions. All subjects who participated in Evaluation 2 had their vision tested at the JSC Clinic and it was determined that they all had either corrected or uncorrected 20/20 vision and none had evidence of color deficiencies.

Apparatus

Testing took place in the ROIL of the NASA Johnson Space Center. The video equipment used in the previous evaluation remained the same except that the DataCube was programmed to display gray scale levels of 8, 16, and 32. Two Lowel Omni 600 watt halogen lamps with a color temperature reading of 3200 degrees Kelvin were used so that the illumination levels could be varied. One of the lamps had spot lighting while the other had flood lighting.

Color chips from the *Munsell Book of Color* were used as the visual stimuli. Color chips selected were the 15 color chips determined by Frederick, Shields, and Kirkpatrick (1977) to be maximally discriminable for both color and direct viewing from a sample of 80 Munsell color chips evaluated. These chips with their respective hues, lightnesses and chromas are listed in Table 1.

The task performed in this evaluation was to determine from the image on the monitor whether two Munsell color chips were either the same or different from one another. The chips were placed on an off-white background that was remotely located away from the subject/monitor area. This background was deemed significant since it closely approximated the color of the payload bay, most thermal insulation, satellites, and structural members that are or will be used in space. Subjects viewed each possible two-chip combination of the 15 chips, with duplicate comparisons (two identical colors side by side) excluded from this study. Therefore, each subject viewed 105 paired-comparisons under each of the three gray scale conditions.

Variables

Two different independent variables were studied in this evaluation: monitor gray scale levels and worksite illumination conditions. The gray scale levels used were 8, 16, and 32 shades of gray. The two illumination levels, as measured at the location of the Munsell chips, consisted of a high illumination level of 16,021 lx (1489 fc) and a low level of 258 lx (24 fc). The high illumination level was approximately equivalent to the lighting conditions of the payload bay of the Shuttle with the sun shining at a high angle. The low illumination condition was approximately equivalent to the lighting conditions of the center of the payload bay of the Shuttle at night when flood lighting is used. Half of the subjects were randomly assigned to each of the two illumination groups. The design can thus be represented by a 2 x 3 two-factor repeated measures design with repeated measures on one factor -- the gray scale levels.

Procedure

The objectives of the evaluation and task instructions were briefly explained to the subjects. Subjects were instructed to respond to each paired-comparison by writing down on a data sheet whether or not the two chips viewed were either the same or different from one another. Each subject performed the 105 paired-comparisons for each of the three gray scale conditions for their respective level of illumination. Color combinations were presented at a rate such that subjects had three seconds in which to make a decision.

Partial counterbalancing was used. Three of the subjects in each group performed the paired-comparison task with the 8 gray scale condition viewed first, the 16 gray scale condition viewed second, and the 32 gray scale condition viewed third. The other three subjects in each group viewed 32 shades of gray first, 16 shades of gray second, and 8 shades of gray last.

Results and discussion

Each subject recorded responses for each of the 105 combinations while viewing all three gray scale levels for either the high or low illumination condition. In total, 315 data points were collected for each of the 12 subjects.

If a subject stated that any paired-comparisons were the same, then this was recorded as an error since none of the Munsell chips were compared to chips of an identical color. Figure 1 depicts the mean subject error-rates plotted across the three gray scale levels for both the high and low illumination conditions. This figure illustrates that the mean error-rates for the 16 gray scale condition was approximately the same (22.4 vs. 24.1) as the 32 gray scale while the mean error-rate for the 8 gray scale condition was substantially higher (37) -- for both illumination conditions.

These data were then statistically analyzed with an ANOVA. The analysis revealed that there was statistical significance ($p = 0.003$) due to the main effect of the different gray scales viewed. The results of the ANOVA did not reveal statistical significance due to the effect of the different illumination levels nor due to interaction effects.

Table 1. Munsell Color Chips used as stimuli.

HUE	COLOR	LIGHTNESS/ CHROMA	HUE	COLOR	LIGHTNESS/ CHROMA
2.5	red	4/14	7.5	green/yellow	6/12
3.75	red	4/14	7.5	green	5/10
8.75	red	max	7.5	green	4/10
6.25	yellow/red	max	7.5	blue/green	4/8
8.75	yellow/red	max	3.75	purple/blue	4/12
2.5	yellow	8/16	10	purple	5/12
2.5	green/yellow	7/12	10	purple	4/12
			5	red/purple	3/10

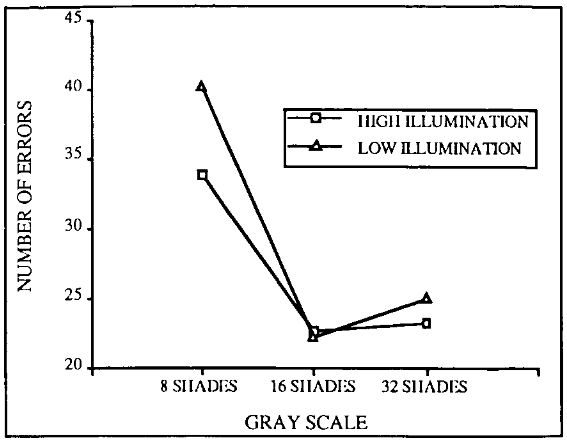


Figure 1. Paired-comparison errors plotted across gray scale and illumination levels

A Newman-Keuls pairwise comparison was then administered to these data. The analysis revealed that the subjects committed significantly more discrimination errors ($p < 0.05$) with the 8 gray scale condition than with either the 16 or 32 gray scale conditions. The Newman-Keuls did not show a significant difference between the error rates of the 16 and 32 gray scale viewing conditions. Since the error rate was significantly worse for viewing with 8 gray scales, it is recommended that for this particular task with monochrome viewing, at least 16 gray scales should be used. This is an interesting finding that may be an addition to the previously stated recommendations (Tannas, 1985) concerning the number of gray scale levels to use. It may well be that for static brightness-level discriminations, gray scale levels as low as 16 are sufficient.

After the study was conducted it was realized that if the most maximally discriminable colors from our test were identified, then these colors would also be discriminable under direct viewing and indirect color video viewing. This would be true because the fifteen color chips used in this study were found to be maximally discriminable for both direct view and indirect color video viewing by Frederick et al., 1975. This information might help establish guidelines concerning the selection of colors which have the least chance of being confused across all three possible viewing methods: direct viewing and both monochrome and color video viewing. An application for this information deals with methods of color coding materials which might be viewed directly by an EVA astronaut or remotely displayed on either monochrome or color video monitors.

When considering viewing under all three methods (direct, color video, and monochrome video), the driving factor is the monochrome condition. A literature review was conducted to determine how many brightness intensity levels are discriminable by an operator. The sources were found to be in disagreement on this matter. For example, NASA-STD-3000 states that no more than 3 brightness intensities be used, Engles and Granda (1975) stated that as many as 4 brightness intensities can be used with some risk of reduced legibility for the dimmer items, Grether and Baker (1972) recommend that no more than 4 levels be used, and Foley and Moray (1987) stated that between 3 and 5 absolute brightness discriminations can be made.

It was beyond the scope of this study to determine which recommendation is correct; therefore, the approach taken was to try to form all of the possible three, four, and five color combinations which had perfect discriminability between them. Since viewing with 8 gray scales was found to be unacceptable, then only the viewing under the 16 and 32 gray scale conditions was included in the analysis. The discrimination-error data across all 12 subjects and both lighting conditions was evaluated with the aid of computer analysis to form the combinations. This analysis yielded 93 three-color combinations, 9 four-color combinations, and 0 five-color combinations that yielded 100% discriminability within each specific color combination for these gray scale levels. The specific combinations may be found in Stuart, Bierschwale, and Smith (1989).

CONCLUSION

The results of this investigation determined that the perceptual discrimination of subjects performing a paired-comparison color-chip task under an 8 shades-of-gray viewing condition was significantly worse than their performance under 16 and 32 shades of gray. A statistically significant difference did not exist between 16 shades of gray and 32 shades of gray for this task. Even though the results in Evaluation 1 demonstrated that there is no perceptible difference between 32 shades of gray and higher gray scale levels, the results obtained in this evaluation may not necessarily be generalizable to the performance of color discrimination tasks using monochrome viewing with 64, 128, and 256 shades of gray since the task was not performed with those levels.

Since the error rate was significantly worse for the 8 shades of gray condition, it is recommended that monitors used for remote monochrome viewing display at least 16 shades of gray if the tasks to be performed are perceptually similar to this task. If the remotely viewed task is more perceptually demanding than the task used in this evaluation, then the displayed gray scale value may need to be even higher. It is recommended that future investigations evaluate the effects of variant gray scale levels on more dynamic telerobotic tasks so that the results obtained will be more generalizable to telerobotic workstation design.

It was of interest to determine the number of three, four, and five-color combinations that had a 100% discriminability rate amongst themselves across both 16 and 32 gray scales and both lighting conditions. These are listed in color combination tables (Stuart et al., 1989) that can be used by systems designers who are faced with the question of how many colors and which specific colors to use for color-coding of tasks which will be performed either EVA or remotely performed and viewed through a monochrome and, or a color monitor. Even though there are other color combinations not evaluated in this study that would have probably produced perfect discriminability under these conditions, these data can eliminate the need to conduct an exhaustive evaluation.

This investigation has provided some insight into an important issue concerning the specific gray scale levels of the monitors to be used for monochrome viewing of a remote inspection task aboard *Freedom* and during later space-based activities. The major result of this study is that it has been determined that 8-gray-scale viewing is unsuitable for monochrome perceptual discrimination tasks. Another result of this study is that it has helped to establish color-coding guidelines concerning the colors which have the least chance of being confused with one another under variant gray scale and illumination conditions. These results have application to both telerobotic workstation monitors and coding of task hardware. The results also have relevance for the performance of remotely viewed tasks in the nuclear and undersea industries.

ACKNOWLEDGEMENTS

Support for this investigation was provided by the National Aeronautics and Space Administration through Contract NAS9-17900 to Lockheed Engineering and Sciences Company.

REFERENCES

Chandlee, G. O., Smith, R. L., and Wheelwright, C. D. (1988). Illumination requirements for operating a space remote manipulator. In W. Karwowski (Ed.), *Ergonomics of hybrid automated systems 1* (pp 241-248). Amsterdam: Elsevier.

Engel, S. E., and Granda, R. E. (1975). *Guidelines for man/display interfaces* (Technical Report TR 00.2720). Poughkeepsie, NY: IBM.

Foley, P. and Moray, N. (1987). Sensation, perception, and systems design. In G. Salvendy (Ed.), *Handbook of human factors* (pp 45-71). New York: John Wiley and Sons.

Frederick, P. N., Shields, N. L., and Kirkpatrick, Jr., M. (1977). *Earth orbital teleoperator visual system evaluation program* (Test Report Number 5). Huntsville, Alabama: Essex Corporation.

Grether, W. F., and Baker, C. A. (1972) Visual presentation of information. In H. P. Van Cott and R. G. Kinkade (Eds.), *Human engineering guide to equipment design* (pp 41-122). Washington, D. C.: American Institutes for Research.

Johnston, D. M. (1968). Target recognition on TV as a function of horizontal resolution and shades of gray. *Human Factors, 10*, 201-210.

Kingdom, F., and Moulden, B. (1986). Digitized images: What type of grey scale should one use? *Perception, 15*, 17-25.

NASA-STD-3000. (1987). *Man-system integration standards.* National Aeronautics and Space Administration.

O'Hara, J. M. (1986). *Telerobotic work system: Space-station truss-structure assembly using a two-arm dextrous manipulator* (Grumman Space Systems Report No. SA-TWS-86-R007). Bethpage, NY: Grumman Space Systems.

Shurtleff, D. A. (1980). *How to make displays legible.* La Mirada, CA: Human Interface Design.

Stuart, M. A., Bierschwale, J. M., and Smith, R. L. (1989). *Effects of variant gray scale levels and workplace illumination levels on operator performance.* (NASA Tech. Report JSC - 23483). Houston, Texas: NASA Lyndon B. Johnson Space Center.

Tannas, Jr., L. E. (1985). Flat-panel display design issues. In L. E. Tannas, Jr. (Ed.), *Flat-panel displays and CRTs* (pp 91-137). New York: Van Nostrand Reinhold.

Troy, E. B., Deutsch, E. S., and Rosenfeld, A. (1973). Gray-level manipulation experiments for texture analysis. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-3*, 91-98.

Woodson, W. E. (1981) *Human Factors Design Handbook.* New York: McGraw-Hill.

N91-20664

SPACE STATION *FREEDOM* COUPLING TASKS: AN EVALUATION OF THEIR SPACE OPERATIONAL COMPATIBILITY

Carlos E. Sampaio, John M. Bierschwale,
Terence F. Fleming, and Mark A. Stuart
Lockheed Engineering and Sciences Company
Houston, Texas

ABSTRACT

The development of Space Station *Freedom* tasks that are compatible with both telerobotic as well as extravehicular activity is a necessary redundancy in order to insure successful day to day operation. One task to be routinely performed aboard *Freedom* will be the changeout of various quick disconnect fluid connectors. In an attempt to resolve these potentially contradictory issues of compatibility, mock-ups of couplings suitable to both extravehicular as well as telerobotic activity have been designed and built. This paper discusses an evaluation performed at the Remote Operator Interaction Laboratory at NASA's Johnson Space Center which assessed the prototype coupling as well as three standard coupling designs. Data collected during manual and telerobotic manipulation of the couplings indicated that the custom coupling was in fact shown to be faster to operate and generally preferred over the standard coupling designs.

INTRODUCTION

After its completion, Space Station *Freedom* will continue to require a great deal of maintenance and support work in order to maintain daily operations. Dextrous manipulators including the Flight Telerobotic Servicer, the Special Purpose Dextrous Manipulator, and the Japanese Experimental Module Fine arm will not only be critical to the performance of these tasks but may actually be the primary system devoted to the execution of many of them.

Among the tasks to be commonly performed will be the coupling and uncoupling of fluid connectors designed to provide remote resupply of liquids and gases in orbit (NASA, 1989). This will be done using various quick disconnect (QD) couplings designed to mate and demate repeatedly without leakage. At present, several designs exist which allow the couplings to be quickly mated and demated by an extravehicular astronaut. While it is critical that these couplings be capable of manipulation by the suited astronaut, it is equally critical that these couplings be capable of successful operation with a telerobotic manipulator in order to reduce the likelihood of these hazardous extravehicular operations in the first place. Consequently, these couplings necessitate a design that is compatible with both modes of operation.

QD coupling designs and methods of actuation can vary widely. The coupling's contents, the amount of pressure it will have to sustain, the amount of flow it will need to accommodate, as well as several other factors all have a bearing on the coupling's final form. Clearly aboard *Freedom*, the varying conditions under which the different QDs operate will necessitate that their designs be different as well. Just as clear, however, is the concern that a proliferance of coupling designs will, at best, often result in uncertainty in a coupling's operation when encountered, and at worst, result in unsuccessful mating or even loss of fluid or pressure as a result of implementing the incorrect coupling process. Although the size and action of the couplings will obviously need to vary, it is preferable that a similar operation concept be shared over the coupling points aboard *Freedom* in order to reduce the likelihood of using the incorrect procedure.

It is widely held that in the vast majority of cases, a task that has been designed to be telerobotically compatible will be compatible with the extravehicular astronaut as well (Newport, 1989). This study, conducted in the Remote Operator Interaction Laboratory (ROIL) at NASA's Johnson Space Center (JSC), evaluated subjects' abilities to mate and demate QD couplings of varying design both telerobotically as well as manually. In a previous study assessing various telerobotic control modes, a manual condition was included as a representation of the optimal performance to strive for in the design of a space glove (Hannaford,1989). Therefore, the manual condition in this study is similarly included as a baseline to reasonably approximate extravehicular activity (EVA).

In collaboration with various telerobotic interface development facilities including the ROIL, Symetrics Inc. has been iteratively designing fluid couplings whose operation is intended to be telerobotically as well as EVA compatible. One of these iteratively designed couplings was among the four coupling designs evaluated in this investigation. Thus the hypothesis of this study proposes that the coupling designed to be telerobotically and EVA compatible will be mated and demated the most quickly and be most preferred subjectively for both the telerobotic as well as manual conditions.

METHOD

Subjects. Four subjects participated voluntarily in this study. In order to minimize learning effects associated with the various systems involved, all subjects had extensive experience with the telerobotic and viewing systems employed in the study. None of the subjects had any experience operating the QD couplings prior to their participation.

Apparatus. Three equipment systems were employed in the ROIL. These are: a telerobotic system, a viewing system, and a task support structure. The telerobotic system consisted of a Kraft force-reflecting master-slave manipulator. The viewing system consisted of three camera views displayed on two 21-inch monitors and one 9-inch monitor. The 21-inch monitors displayed close-up views of the couplings from both front and rear, while the 9-inch monitor displayed an overall view showing the subject the orientation of the manipulator to the task piece. The task support structure consisted of a 72-inch by 48-inch metal frame upon which each coupling was attached one at a time during testing. As demonstrated in Figure 1, the designs of the four couplings included in this study differed, as did their actuation.

Coupling A was demated by grasping the outer sleeve between the two flanges and applying axial force toward the flex hose. It was demated when enough force was applied to overcome the breakout force of the coupling. Mating occurred by aligning the coupling onto the nipple end and applying axial force until the outer sleeve locked back into place. This was the customized coupling designed specifically by Symetrics to be telerobotically and EVA compatible. The flanges of the outer spool-shaped sleeve were designed to be slightly wider than the telerobotic grippers. This allowed some compliance in grappling the fixture while still providing a sufficient brace in order to apply the axial force necessary for demating and mating. Another aspect of coupling A's design which did not exist on the other couplings was a chamfering of the entrance at a 45 degree angle in order to guide the nipple portion into the coupling. It was felt that these compliant features would also lead to enhanced manual operation of the coupling as well.

Coupling B had a very similar mechanism as coupling A. The narrow outer ring was pulled toward the flex hose until the breakout force of the coupling was overcome and the coupling was demated. Mating also occurred by aligning the coupling onto the nipple end and applying axial force until the coupling portion locked back into place.

Demating coupling C required depression of two detents, one on either side of a knurled aluminum ring. Once the detents were depressed, the aluminum ring would slide toward the flex hose and the coupling portion could be pulled away. Mating required aligning the coupling portion onto the nipple end and applying force axially until the detents engaged.

Coupling D had a lever-actuated demating process. The coupling's lever was pushed toward the hard mounted, nipple end. When the lever was pushed to a certain point (approximately 45 degrees), demating automatically occurred. Mating required aligning the coupling and applying axial force onto the nipple end until the lever restored itself to the vertical position.

It is important to note that the task performed in this study does not represent the entire coupling process. The experimental task consisted of, in effect, the soft-latch phase of the coupling process where the coupling is mated or demated but the actual flow of fluid has not been affected. With each of these couplings, the flow of fluid would need to be turned on or off in an additional step not included in the task. That phase of the coupling process would involve the use of an added tool or a modification to the end effector which would drive the coupling into the fully opened or closed position. Since that phase of the process has yet to be defined for Space

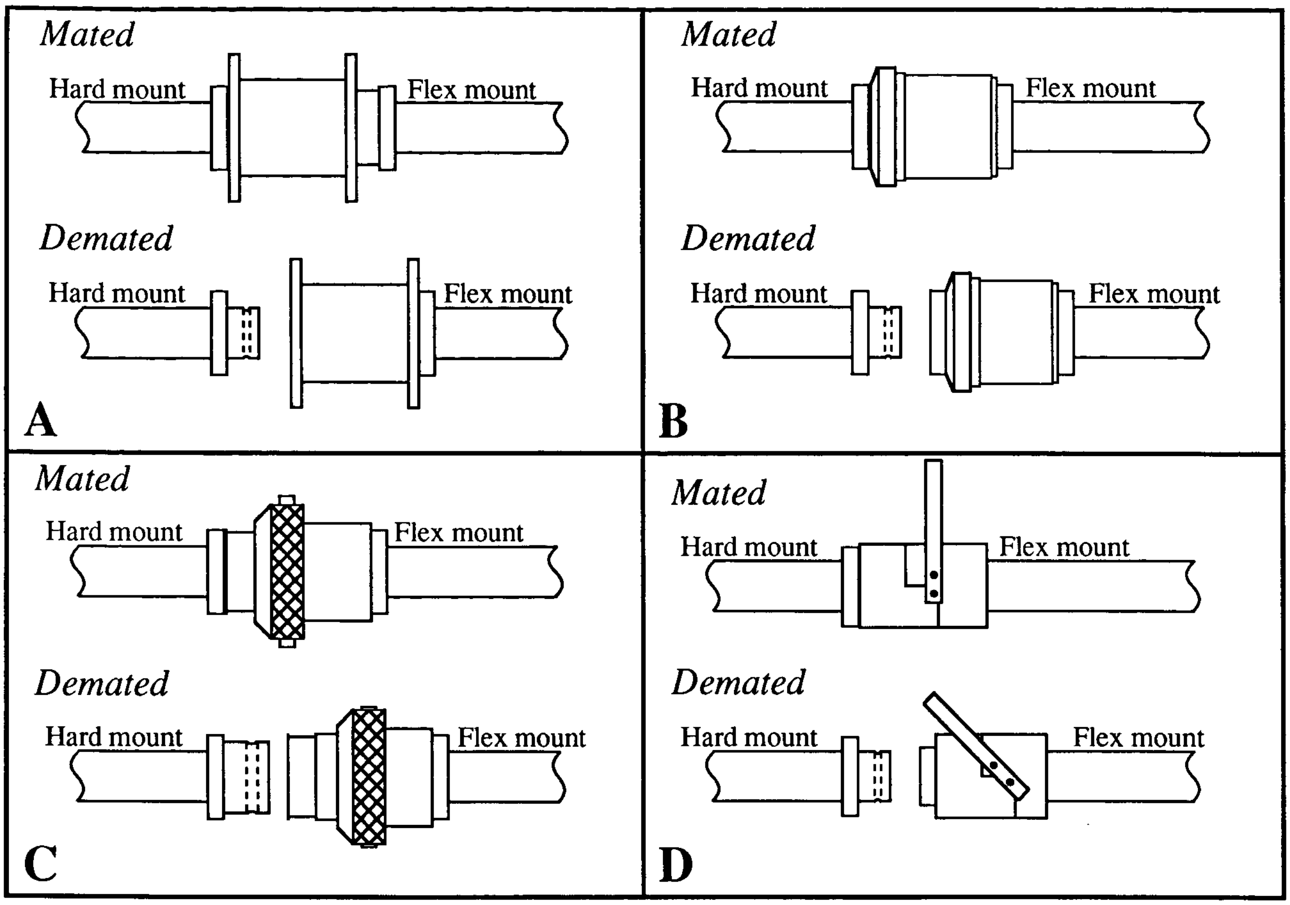


Figure 1. Schematic diagrams of the four couplings evaluated in this study.

Station *Freedom* operations, it was of interest to the experimenters to evaluate the compatibility of the mating and demating components of the task which could be addressed at this time.

Design. This study implemented a 2 modality (manual and telerobotic) by 4 coupling (couplings A, B, C, and D) within subjects design. Modality and coupling sequence was counterbalanced as demonstrated in Table 1.

Procedure. To begin each testing session, subjects were introduced to the purpose and procedure of the study as well as the basic layout of the cameras, task, and robotic system. Since subjects were already familiar with the operation of the robotic and viewing systems employed in the ROIL, no instruction was necessary regarding these aspects of the task.

Subjects began the session by manipulating a coupling either manually or telerobotically depending on their particular counterbalancing sequence. Each coupling was demated and mated three times in each modality. The experimenter kept performance time by means of a hand stopwatch and recorded those times on a data collection sheet where errors were logged as well. An error was counted only if the

Table 1. Counterbalancing sequence for couplings and modality across subjects (M = manual condition, T = telerobotic condition).

Subject	QD Coupling Sequence							
	1		2		3		4	
	Coup. A		Coup. B		Coup. D		Coup. C	
1	M	T	T	M	M	T	T	M
	Coup. B		Coup. C		Coup. A		Coup. D	
2	T	M	M	T	T	M	M	T
	Coup. C		Coup. D		Coup. B		Coup. A	
3	M	T	T	M	M	T	T	M
	Coup. D		Coup. A		Coup. C		Coup. B	
4	T	M	M	T	T	M	M	T

coupling portion was dropped. Timing would then stop while the experimenter reset the coupling and would restart as soon as the subject brought the arm back into motion. Following a set of three trials with each coupling, subjects filled out a short questionnaire with rating scales concerning workload, discomfort, as well as various task related issues. Once all the couplings had been completed, subjects filled out a final questionnaire for each modality where they rated the couplings in comparison to one another.

RESULTS AND DISCUSSION

Analysis of Variance performed on the data showed trends in both the performance as well as subjective data. Table 2 presents the group means for many of the performance and subjective measures. Due to the very few number of errors occurring in any of the trials, analysis of the error data resulted in no significant findings and is not discussed.

It was hypothesized that as a result of the compliant structures built into coupling A, demating and mating it would be faster than other couplings without these structures built into them. Data from the telerobotic trials showed that differences between performance time across the couplings was significant (F (3,3) = 4.372, p <.05). A Duncan's pairwise comparison performed on the data showed that the source of significance came largely from coupling C being significantly slower than all other couplings' performance time. Due primarily to the small variance in the manual condition, differences in performance time did not reach significance for these trials. A Duncan's pairwise comparison on these data, however, did show that performance time for coupling A was significantly faster than coupling C. As anticipated, it appears that for both modalities, coupling A was faster - in some cases significantly faster - to demate and mate than the other couplings.

It was also felt that subjective reactions to the couplings would show preference for the custom coupling in both modalities. The overall rating data were collected on seven point scales with 1 corresponding to "completely acceptable" and 7 corresponding to "completely unacceptable." As shown in Table 2, these data revealed reliable differences, this time for both telerobotic as well as manual ratings. The data regarding the telerobotic preference revealed an F (3,3) = 7.981 with a p <.01. Pairwise comparisons showed that couplings A and B were rated significantly more acceptable than coupling C, while coupling A was significantly more acceptable than coupling D. For the manual ratings the data showed that F (3,3) = 8.007, with p <.01. In this case pairwise comparisons indicated that coupling C was significantly less acceptable than all others. The comparable ratings attributed to couplings A and B appeared the result of their similar mechanisms and operation. The shape of the outer sleeve and coupling A's chamfering was all that varied between the two.

Table 2. Group means for performance and subjective measures.

Measure	**Modality of Operation**							
	Telerobotic				Manual			
	A	B	C	D	A	B	C	D
Perform. Time per Trial (sec.)	66	77	450	168	2.4	3.7	6.4	3.8
Overall Rating (1 to 7)	1.5	2.0	5.3	3.8	1.8	2.5	5.0	2.5
Grip Acceptability (1 to 7)	1.3	2.8	3.5	2.8	1.3	1.8	3.8	1.3
Mental Workload (1 to 10)	3.0	2.5	6.8	4.0	Not Addressed			
Phys. Discomfort (1 to 7)	2.5	1.5	3.8	2.5				

Using the same seven point scale described above, data regarding the acceptability of obtaining the proper grip did not reach significance for the telerobotic condition, although the pairwise comparisons did show that coupling A was rated significantly more acceptable than coupling C. For the manual condition this difference did reach significance, F (3,3) = 5.368, p < .05, with the comparisons among the means indicating that coupling C was significantly less acceptable than all three other couplings.

After the telerobotic trials, data were also collected on mental workload and physical discomfort. Data from a Modified Cooper-Harper mental workload rating scale reached significance, F (3,3) = 3.860, p < .05. The pairwise comparisons showed that couplings A and B were rated significantly less mentally taxing than coupling C. Data from either question addressing physical discomfort did not reach significance although the pairwise comparisons tended to show couplings A and B as less demanding than coupling C. These effects seemed the result of the rather straight-forward mechanism implemented on couplings A and B. Subjects only had to grab and pull to demate couplings A and B, while coupling C required depression of detents on either side of the detention sleeve. This orientation was often very difficult to achieve with the robotic grippers, typically requiring repeated attempts before demate finally occurred. Issues of mental workload and physical discomfort were not addressed after the manual trials due to their very short duration.

CONCLUSION

Of the couplings included in this study, the different operational components resulted in varying reactions from the subjects.

Regarding the demate process, subjects felt coupling D included an attractive feature by requiring little force to demate, achieving it simply by forcing the lever over. However, maintaining control of the coupling portion after demate proved difficult for teleoperation, although somewhat easier for manual operation. Demating coupling C showed that depression of detents is a very delicate operation to perform with the telerobot and to some extent, to perform manually as well. Without some method of fixing the orientation of the detents, it is very difficult to engage both at the same time, particularly with the telerobot. This was compounded by the fact that the depression had to be combined with the axial force necessary to demate. Because of coupling B's small outer ring, demating was at times found to be clumsy with it as well. This was particularly the case for the telerobotic condition, but at times the manual condition was awkward as well.

Mating the couplings proved, on the whole, a far simpler process. Couplings B and D required close alignment which, when met, resulted in a very straight-forward mating process. Coupling C incorporated a longer nipple portion to the coupling. This assisted operation in both modalities by helping to guide the coupling into the mated position when the axial force was applied.

While coupling A did appear the better design in this evaluation, there clearly were facets which could be improved. Although the large flanges on the outer sleeve assisted in mating, they also might allow the telerobot or EVA astronaut to accidentally bump or deactivate the coupling prior to full actuation. Also, the chamfering performed on the entry of the coupling was perhaps angled too far. The 45 degree entrance guided the nipple portion into the coupling, but allowed sufficient misalignment such that the coupling often bound just prior to fully mating. Symetrics has recognized these concerns and has provided the ROIL with a coupling addressing these issues by making two changes in the design. New shorter flanges still allow necessary support for the axial forces required, but greatly reduce the likelihood of accidental deactivation. The entry to the coupling was also chamfered to approximately 30 degrees rather than 45. This assisted in guiding the nipple into the coupling but reduced the potential for binding by lessening the amount of misalignment possible.

The purpose of this study was not to conceive the final coupling design. Rather, it was intended as a step along an iterative process. The newly modified coupling will be included in a series of further controlled as well as subjective evaluations. This is part of ongoing work in the ROIL designed to enhance the overall interface by improving design at both the teleoperator and telerobot ends of the system.

ACKNOWLEDGEMENTS

Support for this investigation was provided by the National Aeronautics and Space Administration through Contract NAS9-17900 to Lockheed Engineering and Sciences Company.

The authors would like to acknowledge John Calvin and Steve Calvin of Symetrics Incorporated as well as Ray Anderson of MacDonnell Douglas Space Systems Company for their assistance in providing materials and information crucial to this evaluation.

REFERENCES

Hannaford, B., Wood, L., Guggisberg, B., McAffee, D., and Zak, H. (1989). *Performance evaluation of a six-axis generalized force-reflecting operator.* (JPL Publication 89-18). Pasadena, California: NASA Jet Propulsion Laboratory.

National Aeronautics and Space Administration (1989). *Interface design considerations for robotic satellite servicers.* (NASA Report JSC-23920). Satellite Servicing Systems Project Office.

Newport, C. (1989, June). *Application of subsea robotics maintenance experience to satellite servicing.* Paper presented at the Satellite Services Workshop IV, NASA Johnson Space Center, Houston, Texas.

N91-20665

GROUND CONTROLLED ROBOTIC ASSEMBLY OPERATIONS FOR SPACE STATION FREEDOM

Joseph C. Parrish
Operations Management Engineer, Space Operations Office
NASA Space Station Freedom Program Office
10701 Parkridge Blvd.
Reston, VA 22091

ABSTRACT

A number of dextrous robotic systems and associated positioning and transportation devices will be available on Space Station Freedom (SSF) to perform assembly tasks that would otherwise need to be performed by extravehicular activity (EVA) crewmembers. The currently planned operating mode for these robotic systems during the assembly phase is teleoperation by intravehicular activity (IVA) crewmembers. While this operating mode is less hazardous and expensive than manned EVA operations, and has insignificant control loop time delays, the amount of IVA time available to support telerobotic operations is much less than the anticipated requirements. Some alternative is needed to allow the robotic systems to perform useful tasks without exhausting the available IVA resources; ground control is one such alternative.

This paper investigates the issues associated with ground control of SSF robotic systems to alleviate onboard crew time availability constraints. Key technical issues include the effect of communication time delays, the need for safe, reliable execution of remote operations, and required modifications to the SSF ground and flight system architecture. This paper addresses time delay compensation techniques such as predictive displays and world model-based force reflection, and describes collision detection and avoidance strategies to ensure the safety of the on-orbit crew, Orbiter, and SSF. Although more time consuming and difficult than IVA controlled teleoperations or manned EVA, ground controlled telerobotic operations offer significant benefits during the SSF assembly phase, and should be considered in assembly planning activities.

NOMENCLATURE

AC	Assembly Complete
APS	Astronaut Positioning System
AWP	Assembly Work Platform
C&TS	Communication and Tracking System
DMS	Data Management System
EVA	Extravehicular Activity
FEL	First Element Launch
FTS	Flight Telerobotic Servicer
IVA	Intravehicular Activity
MMD	MSC Maintenance Depot
MRS	Mobile Remote Servicer
MSC	Mobile Servicing Centre
MSS	Mobile Servicing System
MT	Mobile Transporter
MTC	Man Tended Capability
NASREM	NASA/NBS Standard Reference Model
ORU	Orbit Replaceable Unit
PMC	Permanently Manned Capability
RMS	Remote Manipulator System
SPDM	Special Purpose Dextrous Manipulator
SSCC	Space Station Control Center
SSF	Space Station Freedom
SSRMS	Space Station Remote Manipulator System
TDRSS	Tracking and Data Relay Satellite System
WSGT	White Sands Ground Terminal

INTRODUCTION

Available time for crew intravehicular activity and extravehicular activity during Space Station Freedom assembly is severely constrained. Prior to Permanently Manned Capability (PMC), which occurs on the thirteenth flight of the currently planned 18 flight assembly sequence, the SSF will be visited for 5-7 days by the Space Shuttle at 45-90 day intervals. A crew of five astronauts will be devoted to Station assembly operations during these missions, each working approximately nine hours per day. A total of 36 man-hours of planned EVA will be available for each assembly flight [1]. On many pre-PMC assembly flights, the required crew time to support planned assembly operations exceeds the available resources described above. After PMC, a crew of four astronauts will remain on the Station to perform operations in support of user payloads and core system maintenance. However, planned EVA will only be performed while the Shuttle is present, so EVA time constraints will still influence assembly operations after PMC is achieved. IVA time constraints are significant both before and after PMC; early assembly flights have large requirements for assembly and checkout operations while later assembly missions are performed in parallel with ongoing SSF user payload operations and routine maintenance of the evolving Station.

Robotic systems such as the Flight Telerobotic Servicer (FTS), Mobile Servicing Centre (MSC), and Assembly Work Platform (AWP) will be available on-orbit to augment and reduce crew EVA by providing dextrous manipulation, positioning, and transportation of assembly elements. These devices may be operated from the Orbiter or the Station, depending on the task and the location and availability of workstations. Operation of robotic systems from on-orbit workstations serves to reduce crew EVA time, but places a burden upon crew IVA time. Telerobotic operations take longer to perform than direct (i.e., EVA) manipulation [2]; current estimates used in the SSF program give a factor of three increase in task time for telerobotic operations over EVA. Therefore, the reduction in EVA time provided by the application of robotics must be balanced against the increase in IVA time required to support their operation. As described above, neither IVA nor EVA time will be available in great abundance during the assembly phase.

One potential alternative to IVA control of SSF robotic systems during the assembly phase is ground control. Ground control offers the advantage of relaxed time constraints and the relatively unlimited availability of ground-based human and computational resources. Robotic tasks may be performed while the crew is busy working on other tasks, during periods of crew inactivity such as sleep periods, and even when the SSF is unmanned between pre-PMC Shuttle visits. However, the application of ground control during SSF assembly presents some significant problems which must be addressed. These issues include the effects of communication time delays of up to three seconds, the need for safe and reliable execution of remote operations, and the required modifications to the existing SSF ground and flight system architecture. This paper will trade off the advantages and disadvantages of ground control from an overall system perspective.

Ground control of SSF robotic systems is not in the current program baseline. As mission operations planning continues to identify points in the assembly phase where assembly task requirements exceed crew EVA and IVA time availability, alternative means of accomplishing assembly tasks will need to be investigated. Ground control may serve as the "invisible crewmember" to meet critical SSF assembly objectives.

SSF ASSEMBLY OVERVIEW

The Space Station Freedom is, by far, the most complex system ever deployed and assembled on orbit. The SSF will weigh over 500,000 lbs and span almost 500 ft at Assembly Complete (AC). Current planning involves 18 Space Shuttle flights to deliver assembly elements and pressurized module outfitting, with several more post-PMC logistics flights to support a permanent human presence (Table I).

Table I: SSF Assembly Flight Manifest

Date	Flight	Assembly Elements
3/31/95	MB-1 **(FEL)**	Stbd Inboard PV Power Module, Stbd Truss & Utilities, **AWP**, **APS**, **MT**, **FTS**, Passive Dampers
6/15/95	MB-2	Stbd Truss and Utilities, Stbd Antenna, TCS, Avionics, and Propulsion Pallets
8/30/95	MB-3	Stbd and Port TCS Radiators and Condensors, Stbd Utilities, Stbd Power Mgmt, GN&C, and Payload Support Pallets, Module Support Truss
11/15/95	MB-4	Forward Port Node, Pressurized Docking Adapter, **MRS**, Cupola
1/31/96	MB-5	O_2/N_2 Repress Tanks, Port TCS Pallet, Port and Stbd Truss and Utilities
3/31/96	MB-6	Port Inboard PV Power Module, Port and Stbd Utilities, MT Batteries, Propulsion Pallet
6/15/96	MB-7 **(MTC)**	US Lab Module and Lab Internal Equipment
8/30/96	OF-1	Pressurized Logistics Module, US Lab Internal Equipment, **SPDM**, MMD
11/15/96	MB-8	Aft Port Node, Aft Stbd Node, Node Umbilicals
1/31/97	MB-9	Hab Module and Hab Internal Equipment
3/31/97	OF-2	Pressurized Logistics Module, Hab Internal Equipment, O_2/N_2 Repress Tanks
6/15/97	MB-10 **(PMC)**	Forward Stbd Node, Airlock, Extravehicular Mobility Units (EMUs), Cupola
9/15/97	MB-11	Stbd and Port Outboard PV Power Modules
2/1/98	MB-12	JEM Module and JEM External Equipment
6/15/98	MB-13	ESA Module and ESA External Equipment
9/15/98	MB-14	JEM Exposed Facility 1 and 2, JEM Experiment Logistics Module Pressurized and Exposed Sections
1/31/99	OF-3	Pressurized Logistics Module, Node and Module Internal Equipment, Fluid Mgmt Pallet, Stinger/Resistojet, Payload Support Pallet, External Equipment Upgrades
6/15/99	OF-4 **(AC)**	Pressurized Logistics Module, Module Internal Equipment, Pressurized Docking Adapter Internal & External Equipment Upgrades

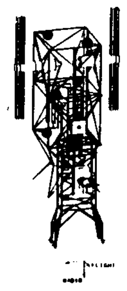


Figure 1a: First Element Launch (FEL)

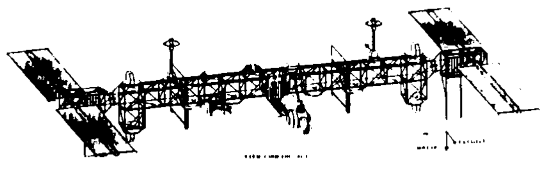


Figure 1b: Man Tended Capability (MTC)

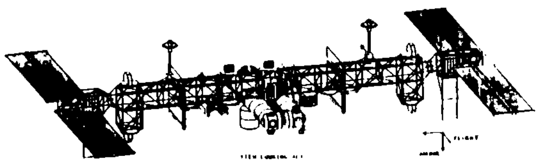


Figure 1c: Permanently Manned Capability (PMC)

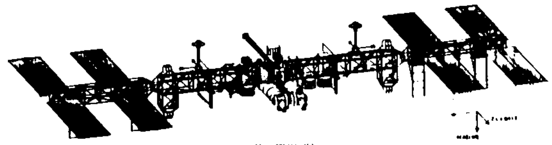


Figure 1d: Assembly Complete (AC)

The assembly phase will last over four years, beginning in early 1995 and ending in mid-1999 [3]. The assembly phase is divided into three subphases, defining logical transitions in ability to support a human crew; these milestones also correlate well with transitions in operational capabilities for SSF robotic systems. The three SSF assembly subphases and their associated crew time availabilities, robotic system capabilities, and assembly task types are described below.

First Element Launch to Man Tended Capability

This subphase (Figure 1a & 1b) spans the first six flights of the current assembly sequence, and is devoted to providing the basic SSF infrastructure (truss, power, avionics, attitude and altitude control, etc.) to support the pressurized modules and attached payloads. Extensive IVA and EVA assembly operations will be performed from the Shuttle during its five to seven day visits to the SSF. No Station-based operations will be conducted during this subphase, and the Station will be unmanned between Shuttle visits. These early assembly missions are highly success-oriented, and an unplanned interruption of assembly operations could potentially threaten the safety of the SSF.

Several SSF robotic systems are delivered during this subphase. The FTS is manifested on the first element launch, and should be available to support dextrous assembly operations by the end of that mission. The Assembly Work Platform (AWP) along with the Mobile Transporter (MT) and its Astronaut Positioning System (APS) is also delivered on the first flight. The Mobile Remote Servicer (MRS), which with the MT comprises the Mobile Servicing Centre, is delivered on the fourth flight. By MTC, the SSF robotic systems can provide payload transportation, positioning, and dextrous manipulation to support assembly operations; the integration of these capabilities will allow many assembly tasks to be performed without the need for manned EVA.

There are two distinct applications of ground control during this subphase. First, ground controlled robotic systems can be used to inspect and checkout assembly tasks between crew EVAs, streamlining the mission timeline and making productive use of non-mission oriented crew activity periods (pre-sleep, sleep, post-sleep, meals, etc.). Minor anomalies might be corrected via ground controlled robotic operations or deferred for later crew controlled IVA or EVA operations. The ground controlled robotic systems may also be applied to setup or breakdown of EVA worksites. The second application is during unmanned phases, where ground controlled robotic systems may be used to complete critical assembly tasks left because of an unplanned interruption of the assembly mission, or to complete non-critical assembly tasks prior to the next Shuttle visit. Obviously, reliability and safety are paramount for these types of operations, since there is no on-orbit crew present to react to a contingency induced by the robotic system.

Man Tended Capability to Permanently Manned Capability

This subphase (Figure 1c) lasts from the seventh through twelfth assembly flights, and is characterized by the delivery and outfitting of the US-provided laboratory and habitation modules. During this subphase, the SSF can support limited operations from within a pressurized shirtsleeve environment. Operational procedures relating to crew safety and pressure differentials between the Shuttle and SSF constrain the crew to the Shuttle until all planned EVA operations on that mission are completed, then SSF IVA operations can commence. Because of this constraint, assembly operations on a particular mission during this subphase tend toward either Shuttle-based external assembly or Station-based internal module outfitting and user operations, but not both. Control of SSF robotic systems will be shared between workstations located in the Shuttle and the SSF. Assembly contingencies during this phase do not directly threaten the safety of the SSF, but may impact the safety of the highly complex pressurized modules, which must be successfully installed within a limited time before subsystem damage occurs.

The only SSF robotic system delivered during this phase is the Special Purpose Dextrous Manipulator (SPDM), which provides dextrous manipulative capabilities similar to the FTS. However, the AWP is "parked" semi-permanently to the truss during this subphase, allowing the MT and MRS to function together as the Mobile Servicing Centre. Transportation and positioning of payloads along the truss is now handled by the MSC.

The two applications of ground control described in the FEL-MTC subphase section above are also valid for this subphase. Assembly operations involving installation of external equipment and pressurized modules are still largely Shuttle-based, with the associated constraints on IVA and EVA crew time. The application of ground controlled inspection and checkout during crew sleep and other periods is still highly beneficial. Again, the potential for an interruption in assembly operations must be addressed. In this subphase of the assembly sequence, the infrastructure for communication between the SSF and ground is complete and can support high bandwidth video downlink and data up/downlink for ground controlled teleoperations while the Station is unmanned. This allows for recovery from some assembly contingencies, and can still be applied to noncritical assembly operations to "get ahead" on the assembly timeline for subsequent missions.

A third application of ground control during this subphase is in support of external assembly operations while the crew devotes its IVA time to internal module outfitting. By eliminating EVA from the timeline, the Shuttle-based crew can gain access to the SSF internal pressurized volume earlier in the mission. Performance of parallel Station-based IVA and ground controlled external assembly operations will require a well-defined choreography between the on-orbit crew and ground controllers, but offers relief from a tight operational constraint.

Permanently Manned Capability to Assembly Complete

This subphase (Figure 1d) lasts from the thirteenth through eighteenth assembly flights, and is characterized by installation of the international pressurized and unpressurized modules and the second set of photovoltaic power modules to bring the Station to its full 75 kW capability. The Shuttle acts primarily as a delivery vehicle during this phase, offloading its cargo to the Station for later Station-based installation. Therefore, robotic assembly operations will be conducted primarily from SSF workstations. However, EVA assembly operations must still be conducted from the Shuttle. SSF IVA operations scenarios during Shuttle-based EVA are still being developed for this subphase.

No new robotic systems are delivered during this subphase. The FTS, SPDM, and MSC will provide dextrous manipulation, positioning, and transportation capabilities to support assembly operations.

It might initially seem that a permanent presence of four crewmembers during this subphase would relieve the IVA time availability constraint. However, user payload operations and core systems maintenance activities will absorb most of the IVA time available, and the four-person crew will probably still operate in a single shift. This leaves large blocks of time per day when no Station assembly operations can be conducted. As described previously, ground controlled telerobotic operations during these periods of non-mission oriented crew activity can be applied to inspect and checkout recent assembly operations or to continue assembly operations along the mission timeline. In summary, opportunities for useful application of ground controlled teleoperations abound throughout the SSF assembly phase.

SSF ROBOTIC SYSTEMS

Successful execution of most SSF assembly tasks requires some combination of transportation, positioning, and dextrous manipulation of assembly elements. The robotic systems described below provide these capabilities, and will be available for use during the assembly phase.

Flight Telerobotic Servicer

The FTS (Figure 2) is a US-developed dextrous manipulator [4]. This device is designed to replace crew EVA, and so attempts to replicate human capabilities. It has two seven-degree-of-freedom manipulator arms, a body providing structure, avionics, utility distribution, and vision capabilities, and a single five-degree-of-freedom positioning and stabilization "leg". These components are arranged in a roughly anthropomorphic configuration. The FTS incorporates force reflection only when control loop time delays are very short (less than or equal to five milliseconds).

The FTS has three basic operating modes during the assembly phase. The first mode, transporter attached, is used when the FTS operates from the Shuttle RMS, SSRMS, or potentially the APS. In this mode, the FTS receives structural and utility support from the transport device, and can operate in areas offering no other means of support. The second mode, fixed base dependent, is used when the FTS operates from a fixed berthing point near the assembly task which provides structural and utility support to the telerobot. In this mode, the FTS is delivered by the transport/positioning device and then operates independently until retrieved for stowage or other operations. The third mode, fixed base umbilical, is similar to fixed base dependent except the telerobot obtains utilities via an umbilical rather than through the berthing point itself.

Prior to the delivery of the SPDM, the FTS is only robotic dextrous manipulator on the SSF. Therefore, the FTS will play a large role in any ground-based telerobotic operations requiring dextrous manipulation. It also offers the advantage of stereographic vision and easily repositionable wrist-mounted cameras, making it a very useful inspection device. It is critical that the FTS be ground-controllable for the applications described in the previous section to be viable.

Mobile Servicing System

The Canadian-built MSS (Figure 3) consists of three ele-

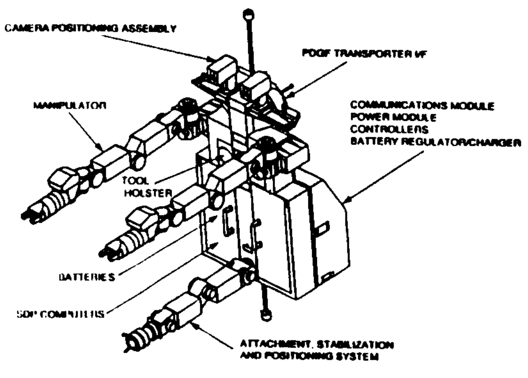


Figure 2: Flight Telerobotic Servicer (FTS)

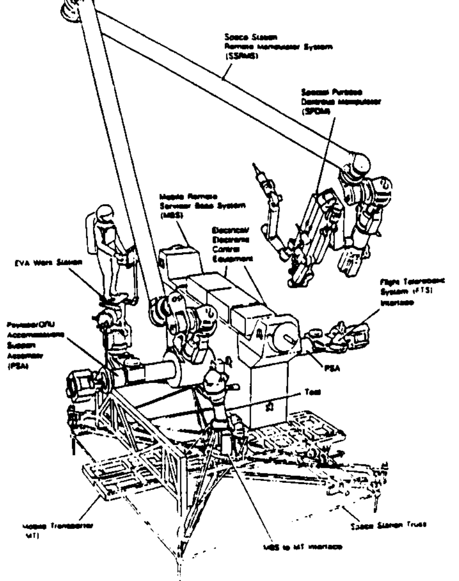

Space Station Remote Manipulator System (SSRMS)
Special Purpose Dextrous Manipulator (SPDM)
Mobile Remote Servicer Base System (MBS)
Electrical Electronic Control Equipment
EVA Work Station
Flight Telerobotic System (FTS) Interface
Payload/ORU Accommodations Support Assembly (PSA)
PSA
Tool
Mobile Transporter (MT)
Space Station Truss
MBS to MT interface

Figure 3: Mobile Servicing System (MSS)

ments: the MSC, the SPDM, and the MSC Maintenance Depot (which performs no manipulative functions and is not considered to be a robotic system) [5]. The MSS provides translation, positioning, and dextrous manipulative capabilities to the SSF.

The MSC, in turn, consists of the MT which provides translation capabilities for the MRS and its payloads, and the MRS which provides payload positioning capabilities using its SSRMS. The MT has an upper and lower base which slide with respect to each other as the MT translates from truss face to truss face. The MT also has the capability to rotate and change planes by independent operation of its upper and lower bases. During early assembly missions (pre-MTC), the MT will be attached to the AWP and will serve as a jig for assembling truss bays. After MTC, the AWP will be detached from the MT, and the MT/MRS combination will move along the truss as a permanently attached unit. The MRS accommodates payloads ranging in size from box-type ORUs to pallets to full-size pressurized modules. It also accommodates the SSRMS, SPDM, and FTS. The SSRMS operates from the MRS, and the dextrous manipulators are stowed on the MRS during delivery to the worksite. Thus, the MSC is both a transportation and positioning device, and must be ground controllable for many of the above-mentioned operational scenarios to be viable.

The SPDM provides dextrous manipulative capabilities to the MSS, and for the purposes of this paper is similar in configuration and characteristics to the FTS. It also operates either from the end of the SSRMS or some fixed berthing point with structure and utility support. The SPDM can play a significant role in ground controlled assembly operations; however, its relatively later arrival in comparison to the FTS makes it a less critical element in a ground controlled telerobotic system. Although it would enhance and augment this capability, it is not crucial that the SPDM be ground controllable.

Assembly Work Platform

The AWP (Figure 4) is a US-built device which serves as a type of scaffolding for assembly of truss structure, routing of utilities, and installation of pallets and other assembly elements on early assembly flights (pre-MTC). The AWP also incorporates the MT for indexing of the truss bays, and the MT's two APSs which serve as analogs to terrestrial cherry-pickers for positioning of EVA astronauts and (potentially) robotic dextrous manipulators. The AWP has many degrees-of-freedom from its various components, but their motions tend to be simply controlled via binary (e.g., on/off, up/down, etc.) commands; therefore, ground control of the AWP would be relatively straightforward. However, the current AWP design has no means of obtaining utilities such as power or data when detached from the Orbiter, so a design modification would have to be made to allow the AWP to operate via ground control between Shuttle visits. The AWP is the assembly base for pre-MTC assembly operations, so it would be essential to provide for ground control of this system to conduct ground-based teleoperations during this assembly subphase.

Shuttle Remote Manipulator System

Until the MSC is available, and in lieu of operations from the APS, the FTS must be transported and positioned by the RMS (Figure 5) [6]. For assembly operations on the first few assembly flights, the RMS is a major payload delivery system. In order to conduct the inspection and checkout operations suggested earlier in this section, the RMS should ground controllable to transport and position the FTS. Use of the RMS under ground control for in-payload bay

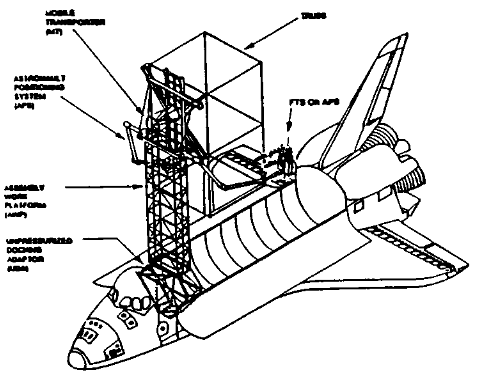

MOBILE TRANSPORTER (MT)
TRUSS
ASTRONAUT POSITIONING SYSTEM (APS)
FTS ON APS
ASSEMBLY WORK PLATFORM (AWP)
UNPRESSURIZED DOCKING ADAPTOR (UDA)

Figure 4: Assembly Work Platform (AWP)

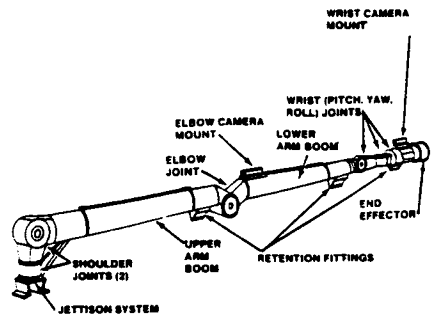


Figure 5: Remote Manipulator System (RMS)

operations such as pallet unberthing and attachment to the truss would probably not be permitted. Nonetheless, the RMS can play a significant role in ground controlled operations on early assembly flights, and should be considered for modification as required to provide this capability.

GROUND CONTROL - ISSUES AND STRATEGIES

Ground control of SSF robotic devices during the assembly phase represents a significant departure from the currently envisioned operational concept. This section discusses potential drawbacks to the use of ground control, and presents several techniques for mitigating these problems.

Communication Time Delays

The space-ground communication link used in the SSF program can induce up to a three second time delay for round trip communications. As shown in Figure 6, communication signals pass from the Space Station Control Center (SSCC) to the SSF via the White Sands Ground Terminal (WSGT) and the Tracking and Data Relay Satellite System (TDRSS). Interestingly, most of the time delay is induced by signal processing on the ground and the SSF, not by the transits to and from geostationary and low earth orbits.

This time delay has a significant impact on teleoperations, where any more than a 0.5 second time delay will cause an operator to adopt a deliberate "move-and-wait" control strategy [7]. Performance of all but the simplest tasks under this type of control is highly inefficient, fatiguing, and error-prone. However, several techniques have been developed to reduce the time delay perceived by the operator, thereby allowing smoother, coordinated execution of task steps.

One of the best-known time delay compensation strategies involves the use of predictive displays, which overlay a

ORIGINAL PAGE IS
OF POOR QUALITY

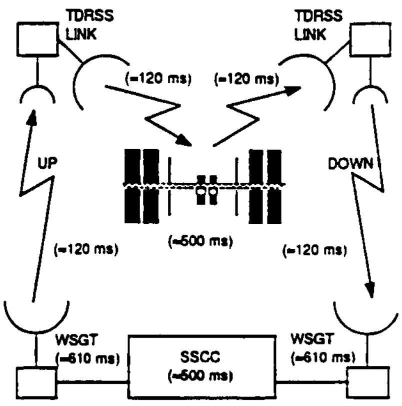


Figure 6: Space-Ground Communication Link

graphical representation of the task prior to the return of the time-delayed video image [8]. These predictive display systems incorporate some knowledge of the manipulator kinematics and dynamics, and may also implement some form of collision detection and avoidance strategy based on a world model of the worksite stored in memory and referenced to the manipulator location and orientation.

Another time delay compensation scheme involves shared or supervisory control of robotic operations, where some portions of the task are automated, relieving the operator of the burden of direct control over all manipulator degrees of freedom [9]. By using shared control, the operator can focus on task goals and monitor the critical functions of the manipulator system (Figure 7). The development of the NASREM and another types of hierarchical control architectures allows shared control to be implemented at various levels of abstraction and can compensate for correspondingly longer time delays.

A novel approach to time delay compensation involves the use of predictive force reflection [10]. This technique may be used independently or in conjunction with a predictive display to define collision free manipulator paths, based on a ground-based world model of the worksite and manipulator system. The operator is then guided away from collisions by a force reaction from the manipulator hand controller. Unlike traditional force reflection techniques, which depend on sensed contact between the manipulator and worksite, predictive force reflection can be preset to a given distance between the manipulator and worksite to warn the operator of impending collisions.

Collision Detection and Avoidance

Clearly, ground-controlled telerobotic operations on SSF pose some risk. The potential for collisions between the robotic system and the SSF and/or Orbiter is probably the largest influence against the adoption of this capability. It is not possible to guarantee collision-free operations even for local, directly viewed teleoperations, much less remote operations under a three second time delay with limited visual feedback. Consequently, the successful application of ground control will depend on the use of some type of collision detection and avoidance strategy. These strategies tend to fall into two types-- sensor-based and world model-based-- as described below.

Sensor-based collision detection and avoidance applies local sensory capabilities (with minimal associated time delay) to compare the location of the manipulator to the worksite and warn the operator of impending contact. The sensors used can range from simple contact switches to infrared, sonic, or electromagnetic field sensors, up to complex machine vision systems. These systems offer the advantage of adaptability to dynamic worksites (which would impair the utility of a static world model-based system), and relatively low computational requirements. However, they are typically expensive and complex in terms of hardware, and less adept at distinguishing between intentional collisions (i.e., between the end effector and ORU grasp point) and unintentional collisions (i.e., between the end effector and ORU support structure).

World model-based collision detection and avoidance techniques involve the use of 3-D computer models of the worksite which are compared to the known position and kinematic configuration of the manipulator. Potential spatial intersections (i.e., collisions) between the worksite model and the manipulator are displayed to the operator. These systems are useful when the bandwidth of the sensory link between the operator and manipulator are small. For the purposes addressed in this paper, the world model would reside on the ground, and manipulator configuration data would be transmitted via the space-ground link. The world model could be as detailed as necessary to support the operation. The major disadvantage of world model-based systems is the dependance on accuracy of the model itself, and the knowledge of the relationship between the manipulator and the task (sometimes known as task registration).

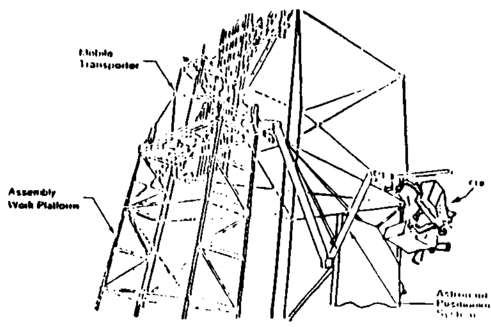


Figure 7: Pallet Installation Using Robotic Systems

Dynamic influences at the worksite such as oscillations or unmodeled equipment may not be accounted for and thus escape detection. Also, offsets between the actual and modeled manipulator due to imperfect task registration can allow collisions to occur without warning.

Operations and Systems Modifications

Implementation of ground control will require some changes to the design and operation of the SSF flight and ground systems. These modifications must be made with minimal impact to existing operations scenarios and system architectures, since ground control is meant to augment, not replace IVA teleoperation when crew time is available.

The implementation of ground control capability is mostly an operational one. The existing SSF robotic systems are commanded via the SSF distributed Data Management System (DMS) and Communication and Tracking System (C&TS). Ground-issued commands can easily be interleaved into these communication links. Therefore, it is largely transparent to the robotic system that it is being commanded remotely rather than locally [11].

The current architecture for collision detection and avoidance for SSF robotic systems relies heavily on crew observation, either directly or through video camera viewing. The difference between such local crew observation and the amount of observational capability available to the ground-based operator should be counteracted by the use of some combination of sensor-based and world model-based systems. The use of these system would also improve the safety of IVA teleoperations by serving as a backup to crew observation.

The existing ground workstations are intended only for simulation and training purposes. Fortunately, these workstations are of high enough fidelity to serve as ground control stations with relatively minor modifications. These modifications will involve integrating the workstations into the space-ground communication link, and incorporating time delay compensation techniques such as predictive displays and world model-based force reflection.

Finally, modifications to the operating modes of robotic systems may require flight equipment modifications. For instance, the AWP is not intended to be used when the Orbiter is not present, and relies on utility provisions from the Orbiter. If ground-based operation of the AWP were to be applied between pre-PMC Orbiter visits, some means for obtaining utilities from the Station would be necessary.

CONCLUSIONS

The current SSF program baseline utilizes ground control for monitoring and limited reconfiguration of onboard systems, and potentially for checkout of robotic systems. No manipulative operations are planned to be controlled from the ground. Given the severe constraints on onboard crew time for IVA and EVA operations throughout the SSF assembly phase, ground control begins to make sense if it can be performed efficiently and safely. The techniques defined above can be applied to increase the operators sense of telepresence and to minimize the risk of collisions between remotely controlled manipulators and the Station and/or Orbiter. More work is needed to determine the specific impacts to SSF ground and flight systems and operations to accommodate this operational mode. Based on this preliminary study, ground control may indeed solve some significant problems associated with the SSF assembly phase.

REFERENCES

1. "Space Station Stage Summary Databook," Space Station Engineering and Integration Contractor, Reston, VA, December, 1989.
2. Akin, David L., "Telerobotic Capabilities for Space Station Operations," SPACE '90, Albuquerque, NM, April, 1990.
3. "Space Station Program Definition and Requirements Section 6: Function and Resource Allocation," NASA SSP 30000 Sec. 6, Rev. E, NASA Space Station Freedom Program Office, Reston, VA, November, 1989.
4. "Flight Telerobotic Servicer (FTS) Design Criteria Document," 87600000005, Rev. N, Martin Marietta Astronautics Group, Denver, CO, March, 1990.
5. "Mobile Servicing System (MSS) Space Segment Contract End Item Specification," SPAR-SS-SG-0276, Issue A, Spar Aerospace Limited, Toronto, Ontario, February, 1989.
6. "Space Shuttle Payload Accommodations, System Description and Design Data - Payload Deployment and Retrieval System," NSTS 07700, Volume XIV, Appendix 8, Revision J, NASA Johnson Space Center, Houston, TX, May, 1988.
7. Sheridan, Thomas B., "Review of Teleoperator Research," NASA Ames Research Lab 20th Annual Conference on Manual Control, Vol. 1, Moffet Field, CA, September, 1984.
8. Delpech, M., and Maurette, M., "Feasibility of Time Delay Compensation for a Space Teleoperation Task," AUTOMATIC CONTROL IN SPACE 1985, Pergamon Press, New York, 1986, pp. 279-286.
9. Parrish, Joseph C., "Opportunities for Space Station Assembly Operations During Crew Absence," JOURNAL OF SPACECRAFT AND ROCKETS, Washington, DC, Vol. 27, No. 3, May-June, 1990, pp. 338-345.
10. Gernhardt, Michael L., Ocean Systems Engineering, Inc., Houston, TX, and Jackson, Eric, International Submarine Engineering, Ltd., Vancouver, BC, personal communication, February, 1990.
11. Hughes, Richard C., National Research Council Canada, Space Station Program, Ottawa, Ontario, personal communication, December 8, 1988.

INTELLIGENT SYSTEMS

TRAJECTORY PLANNING AND SAFETY

N91-20666

A COLLISION DETECTION ALGORITHM for TELEROBOTIC ARMS

Doan Minh Tran
ST Systems Corporation
4400 Forbes Blvd
Lanham, MD 20706

Maureen O'Brien Bartholomew
Code 735
NASA Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

In this paper, the telerobotic manipulator's *collision detection* algorithm is described. Its applied structural model of the world environment and template representation of objects is evaluated. Functional issues that are required for the manipulator to operate in a more complex and realistic environment are discussed.

1 Introduction

Collision detection is the process of detecting imminent collision between moving objects with one another, or a moving object with stationary objects. In a telerobotic environment, detection is concerned with not only collisions between a robot and its surrounding objects but also collisions with itself (i.e., collision between arm's links). Moreover, it involves distinguishing between *unintentional collisions* and *intentional contact* with objects in space. Consider the fact that an end-effector contacting an object such as a coffee mug would not constitute a collision if its goal is to pick up the mug. On the other hand, a collision would occur if the end-effector (or another part of the arm) was to hit any other close-by objects, such as a book located next to the coffee mug. Thus, the collision detection problem requires robot environment awareness on one hand; while on the other, it demands detailed knowledge of objects' characteristics to avoid collisions or to make contact. In other words, collision detection is a process of differentiating between collisions and contacts.

Currently, our collision detection problem is defined by a telerobotic system at NASA Goddard Space Flight Center (GSFC) which consists of two slave Robotics Research Corporation (RRC) arms. Its implementation is part of the safety system which is proposed to serve independently as a redundant monitor of the control system. In addition to redundant collision checks, the safety system performs redundant monitoring of the safety parameters such as velocities, accelerations, torques, and motor currents of the RRC arms. Based on these parameters, it should safely shut down the robot if an attempt is made by the operator to exceed their allowable limits. It also has the capability to override the automatic telerobotic safing functions from the workstation or from other systems. The RRC arm is a seven degree of freedom manipulator, with cylinder shaped links as shown in Figure 1. Movement of the links are rotational in joint-space coordinates. Presently, the arms are in an environment which can be roughly represented as cylindrical or rectangular-shaped objects. In particular, the environment is sparse and well defined with various stands supporting either cameras or Orbital Replacement Units (ORUs). The tasks consists of using the RRC arms to picking, moving, and placing ORU boxes from one location to another. Figure 2 illustrates a typical environment in which the arms operate. It should be noted that the environment is dynamic as the result of ORU boxes being moved by the arms. Therefore, updating of objects' location is necessary.

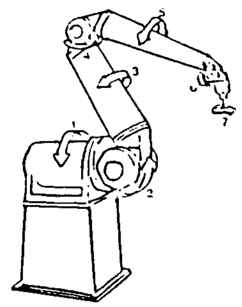


Figure 1: An RRC arm. (Robotics Research Corp. Milford, OH 45150)

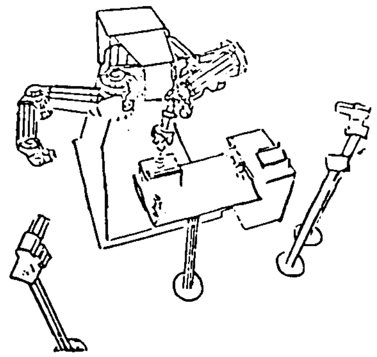


Figure 2: An example of the environment in which the RRC arms operate

2 Collision Detection Algorithm

Having defined the collision problem that may occur in a telerobotic environment and identified the contraints of the manipulators, we will now discuss the collision detection algorithm. In this section, we describe the octree structure for modelling the world and detecting imminent collisions followed by a discussion of an object template representation for distinguishing between intentional contact and unintentional collisions.

2.1 The Octree

In the following paragraph, we describe the octree structure, decomposition of the robot's workspace into regional octree nodes, functional updates for detecting imminent collisions, and node adjustment to reflect environmental changes.

An *octree* is a data structure encoding a space as a tree of either empty nodes or one which consists of a root node and eight disjoint nodes, each of which can be another octree.

The definition of an octree that has just been given illustrates three points. First, an octree of empty nodes is used to indicate object-free space. Second, a node is decomposed into eight sub-nodes (called *octants*) when its contents satisfies some predefined criteria for refining the resolution (this is referred to as the *decomposition rule* which is discussed later). Third, it exhibits recursive inheritance, that is, each octree node is a sub-octree.

Since we are dealing with spatial index octrees, it is convenient to introduce the distinction between them and image representation octrees. In image processing, octree representations are used to define the shape of object by decomposing and representing object's vertices, edges, and planar surfaces as nodes (this is known as a *polytree* [2]). Another way of describing object features is by subdividing the volumes until all leaf nodes are either empty or fully occupied by that object's bounding surface (this is referred to as a *region octree* [7,6,3]). In this sense, the octree node is used to denote an object's shape, as well as for storing object properties such as color and density. A spatial indexing octree, on the other hand, is used to encode the space (in this case, it is the manipulators' workspace). The spatial indexing octree divides the workspace into a set of cubes

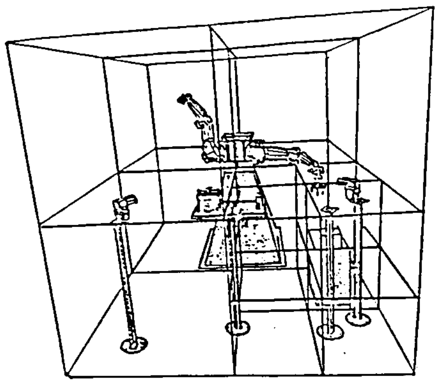
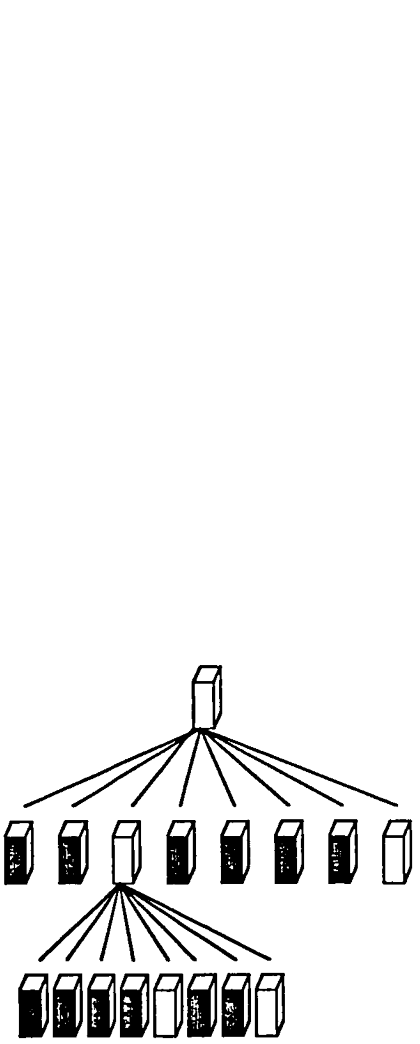


Figure 3: The workspace is partitioned into cubes of various size.

in various volumes. For the sake of consistency, we will refer to the cubes as nodes. Each node, in turn, could either be empty or contain one or more objects. In this context, nodes in an octree denote volumes while their contents hold a list of objects within that regional space [4].

Figure 4 explains our use of the octree. The robot's workspace which consist of a manipulator and various objects can be enclosed in an imaginary cube. This cube is subdivided into eight equal regions, and numbered as shown in Figure 3. In the octree representation, the whole workspace would be denoted as a root node with each sub-region as a sub-node. The workspace in each sub-region can again be subdivided as before and associated with sub-nodes. This process can be repeated until all leaf nodes are either empty or contain no more than three object within it (assuming that the resolution criteria used here allows less than four objects per node). The final octree is shown in Figure 4.

In brief, image representation octrees require a separate octree representation for each object while spatial modeling needs a single octree for encoding all objects. Consequently, spatial index octrees require different methods for splitting nodes than image representation octrees. In the following paragraphs, two methods of decomposing spatial indexing octrees, the *compat-*



□ = Empty Space
■ = The region is filled with objects

Figure 4: The Octree representation of the correspondence workspace.

ibility decomposition rule and *n-objects rule*, are introduced.

In order to grasp the concept of the *compatibility decomposition rule*, we need to understand the notion of *compatible* objects. According to Schaffer and Herb [4], objects or parts of objects (the term *primitive* is used in their paper to covey both object and parts of an object) are compatible if it is impossible for them to collide with each other. For example, an ORU box sitting on top of a stand could not be considered as a collision between the ORU and the stand; thus, these objects are compatible. Mutual compatibleness also exists between any two geometrical abutting links of the manipulator, assuming that the servo level controllers do not allow an angle less than that which would cause the two links to come into contact. In brief, the compatibility decompostion rule dictates the subdivision of an octree node into sub-nodes only if it contains objects that are not mutually compatible. In addition, the compatibility rule appears to involve less octree-updating in a static environment, because updating is only required when a new object is introduced into any region. It is not obvious, however, how compatibility among objects is determined in a dynamic situation. Consider our previous example, compatibility exists between the ORU on top of the stand; but what about picking the ORU box up and then dropping it onto the stand. Clearly, this free-falling/contact action would be considered as a collision between these two objects. Thus, the test of compatibility among objects involves both functional knowledge of objects as well as knowledge of the task being performed.

An alternative decomposition rule might be to subdivide a node if it contains more than "**n**" objects. This rule is similar to the region octree that has been discussed before. Instead of subdiving a node when an object's volume partially fills that region, nodes are split until all leaf nodes containt less than or equal to **n** objects. From the outset, this rule tends to require a lot of updating of the octree as objects are moved from one place to another, regardless of whether they are moved inside or outside of their current regional node. In light of this, the *n-object* decomposition rule is faster than the *compatibility* rule, since it involves no knowledge processing and updating is done only to nodes that contain the moving object(s).

As mentioned before, octree updating is performed everytime objects (i.e., the robot arm, part of the arm, or a box) are displaced. In this context, updating involves both modifying the content of those nodes and checking for collision among objects within a regional node or with neighboring regions.

When objects are moved into another region, node content must be updated. This involves removing the object from its current node and inserting it into the new region. To locate the nearby node, the *neighbor-finding* technique (a traversing technique for locating object's neighboring regions in octree [8]) is used. Neighbor-finding works by first locating the octree node that contains the desired object; then, begin traversing up the tree until a nearest common-ancestor for both the object's node and its desired neighbor node, is found. From that common-ancestor, it descends in a mirror-like direction while ascending the tree. The final stop will be the node that represents the object's neighboring region. It should be noted that, node insertion might change the octree structure. This depends on the decomposition rule that one uses in the algorithm.

According to Boyse [1], detecting a collision for a pair of objects can be done by interference checking of an object's edge with a face of the other or vice-versa. In doing so, one of two things may occur: an object's edge passes through the interior of the other object's face; or it contacts the other object's face boundary. For the former, collision can be detected by determining the locus of each endpoint of the moving edge and examining these loci (space curves) to see whether any intersect the face. In the later case, collision is detected by examining the boundary of the face to see if it intersects the surface generated by the moving edge.

In summary, spatial indexing octrees are useful for detecting imminent collision within the robot's workspace, by encoding the environment as a set of nodes (i.e., cubes) with various volumes. Each node could be subdivided into a set of octants for better resolution (or for manipulation), if its contents satisfy a decomposing criteria. When objects are moved (unless the compatibility decomposition rule is applied and the objects are moved within its current region), the edge-face algorithm is applied to check for collision among objects, and the nodes' structure and/or content are modified to reflect the changes in the environment.

2.2 Template Representation

The problem of distinguishing intentional contact from unintentional collision of objects can be resolved by relying on the system's knowledge of the objects' role with respect to the task. In other words, objects can be categorized from the task as: (1) objects that are manipulated by the telerobot's manipulator and thus come in to contact with it; (2) objects that are caused to collide intentionaly with other objects by the manipulator; or (3) objects which are neither manipulated by the end-effector, nor collided with any other objects. All other types of contacts are interpreted as unintentional collisions.

Imagine an ORU box laying upright on a table. Suppose that in addition to the ORU's position and size, the system also knows it to be a manipulative object (i.e., being picked-up, moved, or placed at other locations by the telerobotic manipulator). The table is viewed as a contacted object. Now, as the end-effector approachs the predefined collision range of the ORU, the collision detection system would assume that the telerobotic's goal is to pick-up that object; thus, it does not view this as an unintented collision (and prohibit any further advancement of the arm). Rather, it allows the arm to proceed at a slower velocity and eventually empowers the end-effector to come into contact with the ORU. The same procedure could be applied to the situation where the manipulator places the ORU on the table; it would not view the contact between these two objects (the ORU and the table) as unintentional collision. However, contact between the telerobotic manipulator and the table *would* be seen as an unintentional collision since the table is viewed by the robot as unmanipulative. Thus, system knowledge of objects' roles enables it to differentiate between intentional contact and unintentional collision. Let us discuss how knowledge of objects' characteristics could be represented internally.

Objects can be defined in terms of their primitive role as: *supporting*, or *manipulating*, or neither; in addition to their shape, size, and position. The *supporting* role denotes an object that can be collided with by another object, providing it is stationary before and during the time of collision. For example, the table illustrated above is defined as having the *supporting* role, and the ORU as having the *manipulating* role. While the *supporting* role allows collision between two objects other than the telerobotic manipulator, the *manipulating* role permits an object (or area of an object) that the end-effector of the robot arm can collide with. One can specify whole or part of object as supportive or manipulative.

More formally, objects can be defined as follows:

```
(object-name
  (primitive-role dimension position))  or
(object-name
  ((component-name
     (primitive-role dimension position))
   (component-name
     (primitive-role dimension position))
   (       :       ))
   dimension position))
```

For example, take a (30"Wx42"Lx38"H) table that is located 60"x50" away from the arm. Its (30"x42"x4") table-top has the *supporting* role (it allows other objects to contact). The remaining components of the table must be protected from contact. It could be described as:

```
(table
  ((table-top
     (supporting (30 42 4) (60 50 34))
   (30 42 38) (60 50 0))
```

Given the object representation above, we intend to solve the problem of differentiating intentional and unintentional collision. Consider, for example, where the manipulator approaches an ORU that is located on a table. Under the current situation, the system predicts an imminent collision: between the arm and the table; and between the end-effector with the ORU. A search of object characteristics in the database indicates that a *supporting* role was assigned to the table, while the ORU object has a *manipulating* role. Based on this information, the system assumes the user intends for the end-effector to contact the ORU but not the table. Thus, it places certain constraints on future movement of the arm. One possible constraint would be for the system to decrease the arm's velocity toward the ORU box; and to inhibit further advancement in the direction of the table.

Another problem in differentiating unintentional collisions from intentional contacts is where the arm holding an object (such as, a ORU box) is about to collide with a stationary object. In the case where the stationary object is supportive (i.e., a table), then it is solvable by assuming that the telerobot's intention is to place the holding object on it. But if the stationary object is manipulative (like another ORU box); then, the event would be declared as an unintentional collision.

Another advantage of this object template representation is, it allows us to logically manipulate parts of an object as unique entities, while it retains the physically inseperable aspects of these components as they make up an object. Hence when an object is moved, all of its components are moved.

In short, knowledge of objects' characteristic (such as *supporting* or *manipulating*) enables the system to predict human operator's intention for the manipulator. Thus, it can inhibit or permit further advances of the arm. Such an approach, however, might lead to collisions of objects due to incorrect assumptions. Nevertheless, these collisions would most likely cause only small physical damage, since the velocity of moving objects are forced by the control system to be very small. A more fail-safe approach to recognizing intentional contact from unintentional collision is to querry the human operator, at the first sight of an imminent collision. However, this would require tedious interaction between the system and the operator, slowing down task performance.

3 Conclusion

In conclusion, a collision detection algorithm for a telerobotic environment must have the ability not only to detect imminent collisions, but also the capability to differentiate between an intentional contact and an unintentional collision. In this paper, we have introduced an octree structure approach to detect imminent collisions. It is a divide-and-conquer algorithm that decomposes the robot workspace into sub-regions. Each sub-region can be empty or occupied with objects. When an object is moved, its edges' intersection with other objects' faces (or vice versa) are calculated in order to detect an imminent collision in the region. On the whole, the spatial index octree approach provides a relatively structured and compact representation, allows a large portion of the workspace to be ignored, and enables real-time updating [5]. However, these advantages depend on the decomposition rules, and those in turn are dictated by the environment and the tasks to be performed.

Once an imminent collision between objects is detected, the system must decide what action should be taken: either stop the telerobot manipulator from further advancement, or set some constraints on the movement of the arm. In order to make this decision, the system must recognize intentional contact and unintentional collision. One solution to this problem is by relying on knowledge of the objects' role with respect to the telerobotic task. For our particular task, objects are defined by their primitive role of either *supporting* or *manipulating*; in addition to their shape, size, and position. Based on this knowledge, the system could infer certain assumptions about the telerobot's intentions as it approaches objects. Consequently, it signals the control system to decreases the arm's movements (in the case of intentional contacts) and outputs warning message, or it inhibits any further advanced of the robot (if it is unintentional collisions). System knowledge of these objects' characteristics help reduce tedious interaction required of the users. However, there is a cost to such approaches. It might lead to collisions of objects due to incorrect assumptions by the system.

4 Discussion

The issue of distinguishing unintentional collisions from intentional contacts has been addressed. What has not been addressed is the issue of interfacing between the collision detection algorithm and other sub-systems within the telerobotic system. In particular, issues that involve describing the world model, database of manipulated and displaced objects by the telerobotic's arm, and handling collisions between objects. Thes problems need to be resolved in order for the manipulator to operate safely and efficiently in a more complex and realistic environment. In the following paragraphs, we address these issues in hope that further research will be conducted to shed some understanding on the problems and their solutions.

The issue of efficient versus effective safeguarding of the operation of the telerobotic manipulator lies partly on the representation of object. In generalizing objects as either solid rectangulars or solid cylinders, we can in effect increase the performance of the safety system due to the simplification of computing objects. By doing so, on the other hand, we have constrained the arm to operate effectively on objects. View a table as a solid rectangular object, for example, we would reduce the computational time describing and detecting collision of objects. But we also inhibit the arm from operating in the space which is under the table. Another classical problem is, how to describe contained objects, such as, a camera in an ORU box. In addition to the redundancy of computing objects, if we differentiate them, we are faced with the problem of processing knowledge that the displacement of the camera might not alter the position of the box but not vice versa. However, if we initialy define the ORU box and its contained camera as one whole object; then any attempt by the telerobotic arm to access the camera will not be possible, since it is viewed as a collision of the arm with the ORU.

Another problem involving safety issues is, when an object is manipulated and displaced by the end-effector. Under this scenario, dynamic or real-time updating of that object's orientation and position are required to detect any imminent collision between the object and other stationary objects or with the arm. It also demands knowledge of which events would cause alternation in object shape while its orientation and position remain fixed. Take an ORU box, for example, where its door is opened by the end-effector. This event requires detection of any collision between the open door with objects (including the arm) that are in its path. It also requires system knowledge that updating the object should only be focused on its shape and not on its position or orientation. In contrast, only the orientation and position of the ORU box are required to be modified, if the arm moves it to another place.

One final issue is one of handling collisions. This problem involves not only when and how to stop moving objects (particularly the telerobotic arm); but also the issue of what information regarding the current environment should be retained prior to system (or components) shut-down must be considered as part of handling a collision. This is due to the fact that, system restarting requires information of the current world.

Acknowledgements

We would like to thank Ravi Kaipa for editing this paper. Thanks are also due to, in alphabetical order: Russ Byrne, Christ Shenton, Scott Swetz, and Janice Tarrant for their helpful comments and suggestions. Research sponsored by NASA Goddard Space Flight Center, Greenbelt, MD.

References

[1] Boyse J.W., *"Interference Detection Among Solids and Surfaces,"* Comm. ACM, vol 22, no. 1, Jan 1979, pp. 3-9.

[2] Carlbom I., Chakravarty I., and Vanderschel D., *"A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects,"* IEEE Comp. Graph. Applications, vol 5, no. 4, Apr 1985, pp. 24-31.

[3] Glassner A.S., *"Space Subdivision for Fast Ray Tracing,"* IEEE Comp. Graph. Application, vol 4, Oct 1984, pp. 15-22.

[4] Herb G.M. and Shaffer C.A., *"A Real Time Robot Collision Avoidance Safety System,"*

[5] Hong T. and Shneier M.O., *"Describing a Robot's Workspace Using a Sequence of Views from a Moving Camera,"* IEEE Trans. Pattern Anal. Machine Intell., vol PAMI-7, no. 6, Nov 1985, pp. 721-726.

[6] Jackins C.L. and Tanimoto S.L., *"Oct-Trees and Their Use in Representing Three-Dimensional Objects,"* Comp. Graph. Image Processing, vol 14, no. 3, Nov 1980, pp. 249-270.

[7] Meagher D., *"Geometric Modeling Using Octree Encoding,"* Comp. Graph. Image Processing, vol 19, no. 2, Jun 1982, pp. 129-147.

[8] Samet H., *"Neighbor Finding in Images Represented by Octrees,"* Comp. Vision, Graphics, and Image Processing, vol 46, 1989, pp. 367-386.

N91-20667

Exact and Explicit Optimal Solutions for Trajectory Planning and Control of Single-Link Flexible-Joint Manipulators*

Guanrong Chen†

Department of Electrical and Computer Engineering

Rice University

Houston, Texas 77251

Abstract. *In this paper, an optimal trajectory planning problem for a single-link flexible-joint manipulator is studied. A global feedback-linearization technique is first applied to formulate the nonlinear inequality-constrained optimization problem in a suitable way. Then, an exact and explicit structural formula for the optimal solution of the problem is derived and the solution is shown to be unique. It turns out that the optimal trajectory planning and control can be done off-line, so that the proposed method is applicable to both theoretical analysis and real-time tele-robotics control engineering.*

1. Introduction

An optimal inequality-constrained trajectory planning problem for a standard single-link flexible-joint manipulator is studied in this paper.

From a structural point of view, a robot arm is a weakly-coupled multi-link mechanical transmission chain. Hence, the study of a single-link manipulator (a unit of a robot arm or an independent mechanism) is of fundamental importance.

It is well known that a trajectory planning problem for a flexible-joint manipulator has a nonlinear model. If we consider such a trajectory planning problem under certain additional optimality criterion, then we will encounter a constrained nonlinear optimization problem. No analytic closed-form optimal solution can be found for such problems in general. However, for a single-link flexible-joint manipulator with single control input, Marino and Spong (1986) shown that a nonlinear feedback configuration can be designed to linearize the non-

* Supported by ONR Contract N00014-88-K-0127.

† The author will join the Department of Electrical Engineering, University of Houston in September, 1990.

linear system globally. The basic idea is, roughly speaking, that one can find a nonlinear feedback to "cancel" the nonlinearity of the system and obtain a linear plant and a linear feedback, leaving the nonlinearity to an explicit transformation. The advantage of this approach is that the final result is exact (no linearization error) after a nonlinear inverse transform. This mathematical technique is of course well known in nonlinear control theory (see, for example, Isidori (1989)). Nevertheless, based on this result, we show in this paper that if we consider a minimum control-energy criterion for the trajectory planning (with inequality-constraints) of such manipulators, then an explicit formulation of the optimal solution for the overall nonlinear constrained optimization problem can be obtained in closed form.

The proposed new approach for obtaining an exact optimal solution explicitly for such an inequality-constrained nonlinear optimization problem is novel in mathematics and very useful in robotics engineering since it provides us an analytic solution before the control process is started, so that no on-line computer is needed in the real-time applications (unless the environment is changing and needs to be adapted), which is sometimes impossible in certain control processing such as in some tele-robotics control in aerospace engineering. Another advantage of closed-form solutions over numerical solutions is the convenience in theoretical analysis of the optimal trajectory planning. Even if in the case that the resultant analytic optimal trajectory cannot be traced by actual control inputs, we know the exact optimal trajectory to be approximated.

This paper is organized as follows: We first describe the optimal trajectory planning problem for a single-link flexible-joint manipulator. Then, we use a standard global feedback-linearization technique to formulate the nonlinear inequality-constrained optimization problem in a suitable way. Based on this mathematical model, we finally give an explicit structural solution for the problem in a closed-form.

2. Description of the Problem

Consider a single-link flexible-joint manipulator as shown in Figure 1.

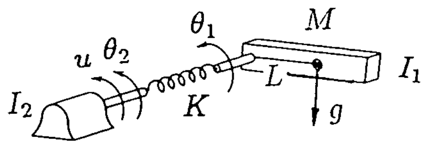


Figure 1. A single-link flexible-joint manipulator

Damping will be ignored in this system for simplicity. The joint is assumed to be of revolute type and the link is assumed to be rigid with inertia I_1 about the axis of rotation. Let θ_1 be the link-angular variable and θ_2 the actuator-shaft angle. Suppose that the rotor inertia of the actuator is I_2. Assume also that the flexible joint is modeled as a linear spring of stiffness K. Then, by the Euler-Lagrange equations we have the following motion equations for this manipulator:

$$\begin{cases} I_1\ddot{\theta}_1 + MgL\sin(\theta_1) + K(\theta_1 - \theta_2) = 0 \\ I_2\ddot{\theta}_2 - K(\theta_1 - \theta_2) = u\,, \end{cases} \tag{1}$$

where M is the total mass of the link, L the distance from the mass-center of the link to the axis of the rotation, g the acceleration constant of gravity, and u the (generalized) force-input applied to the shaft by the actuator.

Since from a mathematical point of view there is no difference between bend and swivel joints (see, for example, Section 5.3 in Nagy and Siegler (1987)), the problem under investigation has rather wide applications.

We will consider an optimal point-to-point trajectory planning problem for this model. To describe the problem more precisely, let $p = p(t)$, $v = v(t)$, $a = a(t)$, $j = j(t)$ be the position, velocity, acceleration, and jerk of the link, respectively, which are functions of the time variable $t \in [0, T]$ for some fixed terminal time $T < \infty$. The first objective is to design a control input $u = u(t)$ to drive the link such that

$$\begin{aligned} &\underline{p}_i \le p(t_i) \le \overline{p}_i, \quad \underline{v}_i \le v(t_i) \le \overline{v}_i, \\ &\underline{a}_i \le a(t_i) \le \overline{a}_i, \quad \underline{j}_i \le j(t_i) \le \overline{j}_i, \end{aligned} \tag{2}$$

$i = 0, 1, \ldots, n$, for some pre-assigned constants

$$\underline{p}_i,\ \overline{p}_i,\ \underline{v}_i,\ \overline{v}_i,\ \underline{a}_i,\ \overline{a}_i,\ \underline{j}_i,\ \overline{j}_i : \quad i = 0, 1, \ldots, n,$$

at the pre-desired time instants

$$0 \le t_0 < t_1 < \cdots < t_n \le T.$$

It can be easily seen from the above trajectory constraints that we will have infinitely many solutions that satisfy the requirements. We want to find an optimal one from them. For this purpose, we consider the problem of controlling the link to satisfy the above trajectory constraints while minimizing certain control-energy to be described precisely later in the next section.

This consideration is especially important when the link is heavy with a large mass M and the control energy (power source) is limited, such as in some aerospace engineering applications.

A direct approach for formulating and solving such a nonlinear inequality-constrained optimization problem does not seem to be easy, unless numerically. However, as mentioned above, numerical solutions are undesirable if analytic solutions can be easily obtained, in particularly, for the purpose of analysis of the control system. In the following two sections, we will first formulate the problem in a suitable way and then derive an closed-form structure for the optimal solution. The resultant optimal solution is actually exact in the sense that no approximation will have been applied.

3. Mathematical Formulation of the Problem

In order to formulate the above-described nonlinear inequality-constrained optimization problem in a suitable way, we first rewrite the motion equations in a state-vector setting and then verify that the resultant nonlinear system satisfies some necessary and sufficient conditions so that it can be linearized globally by a feedback, in the sense that an equivalent but linear closed-loop results. All analyses given in this section are standard in nonlinear systems control theory (see, again, Isidori (1989)) and, in fact, have been done in Marino and Spong (1986) (see, also, Spong and Vidyasagar (1989)) for this particular manipulator model. This technique was also used by Tarn et al (1987).

Let

$$x_1 = \theta_1,\ x_2 = \dot{\theta}_1,\ x_3 = \theta_2,\ x_4 = \dot{\theta}_2,$$

so that equations (1) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u, \tag{3}$$

where $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^\top$, $\mathbf{g}(\mathbf{x}) = [0\ 0\ 0\ I_2^{-1}]^\top$, and

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_2 \\ -I_1^{-1}MgL\sin(x_1) - I^{-1}K(x_1 - x_3) \\ x_4 \\ I_2^{-1}K(x_1 - x_3) \end{bmatrix}.$$

For this nonlinear system, the vector fields $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are smooth, the corresponding Lie brackets $[\mathbf{f}, \mathbf{g}] := \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\mathbf{f} - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{g}$ are given by

$$\begin{aligned} &\{\ \mathbf{g},\ [\mathbf{f},\mathbf{g}],\ [\mathbf{f},[\mathbf{f},\mathbf{g}]],\ [\mathbf{f}[\mathbf{f},[\mathbf{f},\mathbf{g}]]]\ \} \\ &= \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_2^{-1} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ I_2^{-1} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ I_1^{-1}I_2^{-1}K \\ 0 \\ -I_2^{-2}K \end{bmatrix}, \begin{bmatrix} I_1^{-1}I_2^{-1}K \\ 0 \\ -I_2^{-2}K \\ 0 \end{bmatrix} \right\}, \end{aligned}$$

the vector fields

$$\{\ \mathbf{g},\ [\mathbf{f},\mathbf{g}],\ [\mathbf{f},[\mathbf{f},\mathbf{g}]]\ \}$$

are constant and hence form an involutive set, and moreover the vector fields

$$\{\ \mathbf{g},\ [\mathbf{f},\mathbf{g}],\ [\mathbf{f},[\mathbf{f},\mathbf{g}]],\ [\mathbf{f}[\mathbf{f},[\mathbf{f},\mathbf{g}]]]\ \}$$

are linearly independent for all $0 < K, I_1, I_2 < \infty$. Hence, it follows from a result of Su (1981) that the nonlinear system (3) is globally feedback-linearizable, in the sense that an equivalent linear feedback system with an explicit nonlinear inverse transform exists. More precisely, we have the following analysis:

Let $\nabla(\cdot) : R \to R^4$ be the gradient vector of the scalar-valued argument and $\langle\cdot,\cdot\rangle$ a standard inner-product of vector-valued functions. Set $\mathbf{y} = [y_1\ y_2\ y_3\ y_4]^T$ with

$$\begin{cases} y_1 = x_1 \\ y_2 = \langle\nabla(y_1), \mathbf{f}\rangle = x_2 \\ y_3 = \langle\nabla(y_2), \mathbf{f}\rangle = -I_1^{-1}MgL\sin(x_1) \\ \qquad - I_1^{-1}K(x_1 - x_3) \\ y_4 = \langle\nabla(y_3), \mathbf{f}\rangle = -I_1^{-1}MgLx_2\cos(x_1) \\ \qquad - I_1^{-1}K(x_2 - x_4)\,. \end{cases} \tag{4}$$

Then, with the linearizing feedback control of the form

$$u = F(\mathbf{x}) + I_1 I_2 K^{-1} v\,, \tag{5}$$

where

$$F(\mathbf{x}) = I_1^{-1}MgL\sin(x_1)[x_2^2 + I_1^{-1}MgL\cos(x_1) + I_1^{-1}K] + I_1^{-1}K(x_1 - x_3)[(I_1^{-1} + I_2^{-1})K + I_1^{-1}MgL\cos(x_1)]\,,$$

the nonlinear system (3) has been linearized as

$$\dot{\mathbf{y}} = A\mathbf{y} + \mathbf{b}v\,, \tag{6}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

with the following physical meanings:

$y_1 = x_1$ = position of the link

$y_2 = x_2$ = velocity of the link

$y_3 = \dot{y}_2$ = acceleration of the link

$y_4 = \dot{y}_3$ = jerk of the link.

The original nonlinear control system and the equivalent closed-loop and linerized feedback configuration of the overall system are shown and compared in Figure 2 below.

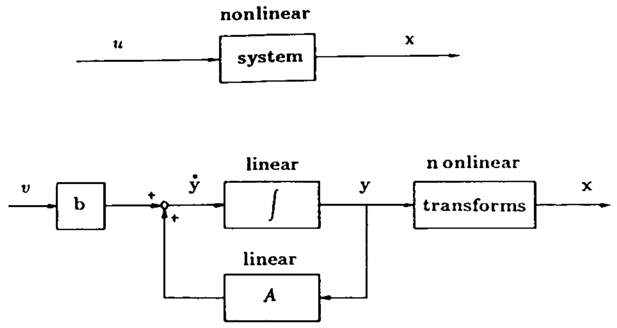


Figure 2. Equivalent Feedback Loops

Here, it is important to point out that if we only consider the trajectory planning (constraints (2)), then the systems shown in Figure 2 are equivalent in the sense that if the trajectory of the linearized feedback system can be controlled to satisfy the constraints (2), then the same can be done for the original nonlinear system by inverting the nonlinear transform (4). For this reason, from now on we can leave the original nonlinear system and work on the linear system (6) together with the nonlinear feedback (5) instead.

Note that in the linearized feedback system, v is the only external and active control input, as can be seen from Figure 2 above. Hence, in the study of the trajectory planning for the linearized feedback system instead of the original nonlinear system, we may consider to minimize the total control-energy of this executive input v. Based on this point of view, we formulate an optimal trajectory planning problem as follows:

Problem:

$$\min_{v\in L_2(0,T)} \int_0^T v^2(t)dt \tag{7a}$$

subject to the linear system

$$\dot{\mathbf{y}} = A\mathbf{y} + \mathbf{b}v \tag{7b}$$

and the trajectory constraints

$$\begin{aligned} &\underline{p}_i \le p(t_i) \le \overline{p}_i, \quad \underline{v}_i \le v(t_i) \le \overline{v}_i, \\ &\underline{a}_i \le a(t_i) \le \overline{a}_i, \quad \underline{j}_i \le j(t_i) \le \overline{j}_i, \end{aligned} \tag{7c}$$

$i = 0, 1, \ldots, n$, *where* $0 \le t_0 < t_1 < \cdots < t_n \le T < \infty$.

Here, $L_2(0,T)$ denotes the standard Hilbert space of square-integrable real-valued functions defined on the time-interval $[0,T]$. Once we have solved the linear constrained optimization problem (7a-c) for optimal v^* and $\mathbf{y}^* = [y_1^* \; y_2^* \; y_3^* \; y_4^*]^\top$, we obtain an optimal solution for the original problem from the inversion of (4); namely, from the following formulas:

$$\begin{cases} \mathbf{x}^* = \begin{bmatrix} y_1^* \\ y_2^* \\ y_1^* + I_1K^{-1}[y_3^* + I_1^{-1}MgL\sin(y_1^*)] \\ y_2^* + I_1K^{-1}[y_4^* + I_1^{-1}MgLy_2^*\cos(y_1^*)] \end{bmatrix} \\ u^* = F(\mathbf{x}^*) + I_1I_2K^{-1}v^*. \end{cases} \quad (8)$$

In the next section, we will show a closed-form structure for the optimal solution of the inequality-constrained optimization problem (7a-c).

4. Closed-Form Optimal Solution

In this section, we show the closed-form structure of the optimal solution for the inequality-constrained optimal trajectory planning problem formulated in (7a-c) above. As a result, the optimal solution for the original nonlinear inequality-constrained optimization problem turns out to be exact in an explicit closed-form.

In order to state our result precisely, we need some new notations. In addition to the notations used before, set the matrix-valued exponential function

$$e^{(t-\tau)\begin{bmatrix} 0 & I \\ A^\top A & A - A^\top \end{bmatrix}} := \begin{bmatrix} E_1(t-\tau) & E_2(t-\tau) \\ E_3(t-\tau) & E_4(t-\tau) \end{bmatrix},$$

$0 \le \tau, t \le T$, in which each submatrix $E_i(t-\tau), i = 1,2,3,4$, is a 4×4 block. Then, using the notation $\mathbf{1} := [1 \; 1 \; 1 \; 1]^\top$, define

$$\mathbf{h}(t-\tau) = E_2(t-\tau)\mathbf{1}$$

and

$$\mathbf{h}_+(t-\tau) = \begin{cases} E_2(t-\tau)\mathbf{1} & t \ge \tau \\ 0 & t < \tau. \end{cases}$$

Moreover, let

$$\dot{\mathbf{h}}(t-\tau) = \frac{\partial}{\partial t}\mathbf{h}(t-\tau)$$

and

$$\dot{\mathbf{h}}_+(t-\tau) = \begin{cases} \frac{\partial}{\partial t}\mathbf{h}(t-\tau) & t \ge \tau \\ 0 & t < \tau. \end{cases}$$

Our main result can now be stated as follows:

Theorem. *The optimal solution for Problem (7a-c) is given by*

$$\begin{cases} \mathbf{y}^*(t) = C_{01}\mathbf{h}_+(t-t_0) + C_{02}\dot{\mathbf{h}}_+(t-t_0) \\ \qquad\qquad + \sum_{i=1}^{n-1} C_i\mathbf{h}_+(t-t_i) \\ v^*(t) = \dot{y}_4^*(t), \end{cases} \quad (9)$$

where $C_{01}, C_{02}, C_i, i = 1,\ldots,n-1$, are all 4×4 diagonal constant matrices which are uniquely determined by the trajectory constraints (7c) from the given data set

$$\underline{p}_i,\ \overline{p}_i,\ \underline{v}_i,\ \overline{v}_i,\ \underline{a}_i,\ \overline{a}_i,\ \underline{j}_i,\ \overline{j}_i,\ : \quad i = 0,1,\ldots,n. \quad (10)$$

Consequently, the optimal solution $(u^*, \mathbf{x}^*)$ for the original problem is obtained via (8) from the optimal solution $(v^*, \mathbf{y}^*)$ given by (9).

We remark that the determination of the constant coefficient matrices of $\mathbf{y}^*(t)$ is simple, which can be done easily by using any standard quadratic programming algorithm even before the manipulator control processing is started, so that no on-line computer is needed for this optimal nonlinear trajectory planning problem unless adaptive control is necessary. More precisely, we demonstrate this procedure as follows: First, we observe that the minimization problem

$$\min_{v \in L_2(0,T)} \int_0^T v^2(t)dt$$

is equivalent to either

$$\min_{v \in L_2(0,T)} \int_0^T [\mathbf{b}v]^\top[\mathbf{b}v]dt$$

or

$$\min_{\mathbf{y} \in H_1(0,T)} \int_0^T [\dot{\mathbf{y}} - A\mathbf{y}]^\top[\dot{\mathbf{y}} - A\mathbf{y}]dt, \quad (11)$$

where $H_1(0,T)$ is the standard first order Sobolev space. If we can solve the minimization problem (11) for $\mathbf{y}^*$, then we can find the optimal solution $v^* = \dot{y}_4^*$ (see (6)). Secondly, we notice that the minimization problem (11) together with the inequality-constraints (10) can be reformulated as the following quadratic programming problem:

$$\min_{C_{01},C_{02},\{C_i\}} \mathbf{C}^T\mathbf{W}\mathbf{C} \qquad (12)$$

subject to

$$\underline{p}_i \le p(t_i) \le \overline{p}_i, \quad \underline{v}_i \le v(t_i) \le \overline{v}_i,$$
$$\underline{a}_i \le a(t_i) \le \overline{a}_i, \quad \underline{j}_i \le j(t_i) \le \overline{j}_i,$$

$i = 0, 1, \ldots, n$, where $\mathbf{C}$ is a constant vector consisting of all elements of the diagonal matrices $C_{01}, C_{02}, C_1, \ldots, C_n$ and $\mathbf{W}$ is a constant matrix consisting of the integrations of all elements of the functions $\mathbf{h}_+$ and $\dot{\mathbf{h}}_+$. Both $\mathbf{C}$ and $\mathbf{W}$ have simple explicit expressions as can be easily seen and derived from formulas (9) and (11). This standard quadratic programming problem can be solved by some existing computer routines, which will provide us the unique optimal solution.

5. Conclusions

In this paper, we have studied an optimal trajectory planning problem of a standard single-link flexible-joint manipulator. We first used a standard feedback-linearization technique to formulate the nonlinear inequality-constrained optimization as a minimum control-energy problem. Then, we have derived an exact structural formula in closed-form for the optimal solution of the problem and showed that this solution is unique. The proposed approach is applicable to both theoretical analysis and real-time robotics control engineering.

REFERENCES

[1] A.Isidori (1989): *Nonlinear Control Systems: An Introduction*, second edition, Springer-Verlag, New York.

[2] R.W.Marino and M.W.Spong (1986): Nonlinear control techniques for flexible joint manipulators: A single link case study, Proc. 1986 IEEE Conf. Robotics and Automation, San Francisco, April 1986, pp.1030-1036.

[3] F.N-Nagy and A.Siegler (1987): *Engineering Foundations of Robotics*, Prentice-Hall, Englewood Cliffs, New Jersey.

[4] M.W.Spong and M.Vidyasagar (1989): *Robot Dynamics and Control*, Wiley, New York.

[5] R.Su (1981): On the linear equivalents of nonlinear systems, Sys. Contr. Letts., Vol.2, pp.48-52.

[6] T.J.Tarn, A.K.Bejczy, and X.Yun (1987): Nonlinear feedback control of multiple robot arms, Proc. NASA Workshop on Space Telerobotics, Jan. 20-22, 1987, Pasadena, CA.

N91-20668

ISSUES ASSOCIATED WITH ESTABLISHING CONTROL ZONES FOR INTERNATIONAL SPACE OPERATIONS

Blair A. Nader
Kumar Krishen, Ph.D.
NASA Lyndon B. Johnson Space Center
Houston, Texas 77058

Abstract

Cooperative missions in Earth orbit can be facilitated by developing a strategy to regulate the manner in which vehicles interact in orbit. One means of implementing such a strategy is to utilize a control zones technique that assigns different types of orbital operations to specific regions of space surrounding a vehicle. This paper considers the issues associated with developing a control zones technique to regulate the interactions of spacecraft in proximity to a manned vehicle. It includes discussion of technical and planning issues, flight hardware and software issues, mission management parameters, and other constraints. It addresses manned and unmanned vehicle operations, and manual versus automated flight control. A review of the strategies utilized by the Apollo Soyuz Test Project and the Space Station Freedom Program is also presented.

Introduction

To date, space operations have been conducted in the absence of a large body of international regulations. While some guidelines have been defined in agreements such as the 1967 Outer Space Treaty, each nation has operated according to its own priorities and capabilities. As the number of space-faring nations and orbiting spacecraft increases, it seems desirable to develop an international strategy to coordinate, monitor, and control the interactions of spacecraft in orbit. Successful strategies will facilitate cooperative operations while supporting each nation's goals and objectives in space. The potential benefits of such a strategy include reductions in future program costs and increases in mission success and safety through the standardization of space operations and equipment; increased safety through development of a coordinated collision avoidance strategy for active spacecraft; and the establishment of a basis for legal and economic compensation agreements.

Any traffic management concept should address a number of general requirements. First, the concept should allow for standardized mission planning and operations. To facilitate this, the routine need for long lead time preparation prior to execution of the mission should be avoided. Second, the concept should permit standardized flight and ground crew planning and operations. This standardization will simplify training and day-to-day activity planning. Third, the concept should allow for early definition of requirements for communications, tracking, telemetry, and command and control. Finally, any plan for coordinating space operations should provide for collision avoidance between spacecraft and hold disturbances and contamination from thruster firings to a reasonable level.

There are many ways to meet the requirements outlined above. One means is to utilize a control zones technique that assigns different types of orbital operations to specific regions of space surrounding a vehicle[1]. Such a strategy offers the advantage of clearly delineating the responsibility of both vehicles as a function of their relative positions, velocities, and time. It is unambiguous because these quantities can easily be determined onboard the spacecraft or on the ground using existing technology. While zone-based strategies can be utilized to regulate a wide range of orbital operations, this paper only considers the issues associated with developing a control zones technique to regulate the interactions of spacecraft in proximity to a <u>manned</u> vehicle. However, it should be noted that many of these same issues will apply when expanding the control zones concept to longer ranges and more classes of spacecraft.

This paper outlines a set of items that should be considered when developing international standards for traffic management using a zone-based technique. It then discusses each of these items in terms of the major issues that will influence its standardization. Cost implications are also discussed, where appropriate.

It should be noted that this paper does not attempt to address the full range of policy-related questions, such as defining the legal basis that nations have for establishing some form of authority over a region of outer space. It assumes that such questions will be answered elsewhere. Nor does it seek to sell the worth of a control zones strategy. Rather, it assumes that the international community will recognize the benefits of such a strategy for at least one class of orbital vehicles (e.g., space stations) and develop an appropriate set of international standards.

Definitions

The following terms will be utilized throughout this paper and are collected here to facilitate understanding of the issues and provide easy reference.

- <u>Rendezvous target</u> - the vehicle that one is attempting to rendezvous with.
- <u>Control authority</u> - the authority to make major decisions on the conduct of a mission, such as aborting the mission.
- <u>Flight crew</u> - the personnel onboard the manned base.
- <u>Ground crew</u> - the personnel on the ground who support the premission preparation and real-time execution of the manned base's mission.
- <u>Vehicle classes</u> - vehicles that possess the same basic characteristics and fulfill the same basic mission. For example, the Soviet Shuttle Buran and the American Shuttle Atlantis belong to the same vehicle class; both are manned vehicles that ferry crews and supplies into orbit.
- <u>Teleoperated vehicles</u> - unmanned vehicles that are remotely piloted during all or part of their nominal trajectory. For example, NASA's orbital maneuvering vehicle will utilize a ground-based pilot to perform docking operations with the target spacecraft.
- <u>Autonomous vehicles</u> - unmanned vehicles that execute rendezvous, proximity operations, and docking through the use of automated flight control techniques and do not nominally require a remote pilot.
- <u>Zonal authority</u> - the authority granted to a manned base within a specified region of space by the international community. It includes the rules of operation within such a zone, as developed by the base vehicle's controlling nation, program office, or control center

and agreed to by the international community.

- Control zones - the regions of space in which zonal authority is exercised. For Space Station Freedom, this is called the "command and control zone" (Figure 1).
- Manned base - a manned spacecraft that has been allocated control zones.
- Zonal compliance - meeting the requirements of a given control zone.
- Transient vehicles - those spacecraft that enter a given control zone, but do not nominally plan to interact with the manned base (e.g., operational satellites).
- Mission management parameters - For the purpose of this paper, these are defined as data that enable ground controllers and onboard crewmembers to monitor mission progress and make decisions in real time. Included in these data are the nominal mission plan, preflight determined contingency plans, real-time status of safety critical vehicle systems, vehicle state vector data, etc.
- Interacting vehicle - any spacecraft that enters the manned base's control zone.
- Zone activation period - the period of time when the manned base may exercise its authority over its control zone.

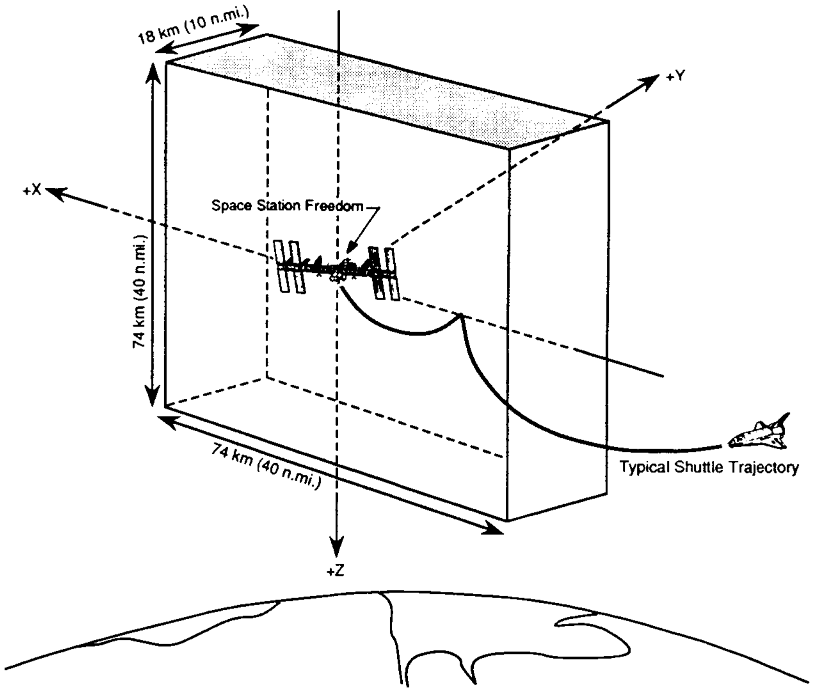


Fig. 1 Space Station Freedom's command and control zone.

- Dynamic control - to actively pilot a vehicle.
- Berthing - the linkup of one orbiting object with another, wherein the closing energy is provided in a closely controlled fashion by an intermediate mechanism attached between the two[2]. This mechanism is typically a remote manipulator, such as the Space Shuttle remote manipulator system.
- Observable parameters - data that can be obtained through observation of the interacting vehicle by active (i.e., radar) means or by passive means (eyeball) from the manned base without the use of telemetry.

Historical Precedents

This section summarizes the traffic management-related elements of Space Station Freedom and the Apollo Soyuz Test Project in order to outline the historical precedents for the remainder of the paper. The references cited can provide more information to the interested reader.

Space Station Freedom

Space Station Freedom plans to implement a limited control zones strategy for regulating its interactions with other spacecraft. This strategy assumes a command and control zone as shown in Figure 1. The regulations for this zone designate the ground as the primary control authority, until the vehicle enters the control zone. Then, Freedom becomes the primary control authority[3]. While the specific regulations vary as a function of vehicle class, Freedom has the authority to "wave off" cooperating manned or unmanned vehicles operating within the control zone[3]. For unmanned vehicles, Freedom will "exercise dynamic control" and have "hazard critical systems monitoring/command capability"[3] while they are inside the control zone. The unmanned vehicle's ground control center will serve as a backup for Freedom's dynamic control and also monitor the full set of systems parameters. In the event of a communications failure between Freedom and the vehicle, control will revert to the ground. Alternatively, when interacting with manned Space Shuttle Orbiters, Freedom will have two-way voice communication[3], but will rely on the ground for trajectory and systems monitoring. It should be noted that regulations have not yet been developed for manned vehicles other than U.S. Orbiters (e.g., Hermes or Buran).

Freedom is implementing the necessary communications capabilities to support these regulations. At the same time, other vehicles such as the Space Shuttle Program is developing the necessary Freedom-compatible interfaces.

Apollo Soyuz Test Project

The Apollo Soyuz Test Project was among the first instances of cooperative international space operations. Its purpose was to dock two manned spacecraft in low Earth orbit: the American Apollo and the Soviet Soyuz. The test project did not utilize a control zones strategy as such. Rather, it relied on a set of mission-unique flight rules that were jointly agreed upon prior to the mission. However, these mission rules[4] had the same effect (i.e., to monitor and control the interactions of two spacecraft in proximity to each other).

Apollo began monitoring the relative trajectories upon sensor acquisition. Soyuz served as the rendezvous target while Apollo performed the actual rendezvous maneuvers. The Apollo and Soyuz ground control centers were the control authorities for most of the mission. However, the spacecraft commanders exercised controlling authority during the docking phase. They also exercised authority if communications were lost with the ground, or in contingency situations that required rapid responses. The ground monitored vehicle health and trajectory, computed the long-range rendezvous maneuvers, and coordinated mission execution. Once within its sensor range, Apollo computed the necessary rendezvous maneuvers (though the ground retained primary control authority). Apollo and Soyuz could communicate between themselves by voice link and thereby exchange relevant data, but neither could monitor the other's telemetry.

Items To Be Considered When Developing International Standards

Development of a zone-based traffic management strategy will require the international community to agree upon a set of standards that reflect a wide range of technical disciplines and issues. Such standards must be specific enough to be useful for near-term missions, but flexible enough to serve as a basis for long-term coordination and cooperation in space. This section outlines eight areas that should be considered in developing these standards.

1. First, the community should determine the classes of spacecraft that should be assigned control zones. For example, will only space stations have them, or will shorter duration orbital missions (e.g., space shuttles) also benefit from having such zones assigned to them?

2. Second, the type of vehicles that must comply with such zones should be defined. For example, must satellites whose orbits occasionally cross a manned base's zone comply with communications requirements for that zone?

3. Third, the size of zones allotted to each class of manned base should be decided. For example, Space Station Freedom currently has a control zone that extends +37 km (20 n.mi.) horizontally and +37 km (20 n.mi.) vertically (Figure 1). This zone is +9 km (5 n.mi.) in the out-of-plane dimension.

4. Next, the community should agree upon the regulations that apply to vehicles operating within a control zone. By analogy, these are similar to laws governing commercial air traffic. They regulate which vehicle is in control at a given time, the approach corridors to be utilized, monitoring requirements, etc.

5. The community should define the parameters that a manned base will monitor within its control zone. For example, will they be limited to trajectory data or will they include safety-critical systems data for the interacting vehicle?

6. The duration of each zone's activation should be determined. For example, should a vehicle's control zone be active continuously, or should it only be active during particular mission phases or operations?

7. The performance parameters necessary to ensure compatibility of communications and telemetry should be considered. These parameters can include the frequency of operation, polarization, spatial coverage modulation and demodulation protocols, and system operational modes.

8. The international community should specify the tracking parameters and

measurement accuracies necessary to assure tracking system compatibility.

While the list just presented is not comprehensive, it is thought to represent the scope of the problem. Accordingly, each item on this list will now be addressed in a separate section where the relevant issues are discussed in detail. Discussion will include its technical and planning aspects, as well as concerns with flight hardware and software, mission management parameters, etc. These issues, in turn, can be utilized to identify follow-on studies that will ultimately result in a set of international standards. Note that a summary of both the items and their related issues is presented in Table 1.

Discussion of the Issues

1. Classes of manned spacecraft to be assigned control zones

One advantage of control zones is that they provide a framework in which to organize and coordinate on-orbit operations between programs and nations. However, it must be noted that joint missions have been conducted successfully in the past without a control zones strategy (e.g., Apollo Soyuz Test Project). In such cases, negotiations are conducted between the nations or parties involved to determine the physical interfaces, flight rules, constraints on mission design, and other details.

Table 1 Summary of standardization items and their related issues

Standardization	Related issues*
1. Classes of manned vehicles to be assigned control zones	• Number of interactions = f(class of manned vehicle) • Vehicle design = f(cost to modify) • Mission design, mission management data = f(standardization)
2. Classes of vehicles that must comply with control zones	• Apply zone to all vehicles = f(cost to implement, workarounds) • Frequency of interaction = f(cost of mission-unique planning) • Controllability = f(manned, unmanned, autonomous) • Hardware & software = f(vehicle design, zonal regulations, zone size) • Planning, mission management data, training = f(interaction, contingency planning, zonal regulations)
3. Size of zone allotted to each manned base	• Safety = f(vehicle class, trajectory, relative velocities, evasive maneuvers) • Base's altitude = f(value of orbit, population) • Hardware & software, planning, mission management data, training = f(zone size, hardware limitations)
4. Regulations that apply within control zones	• Apply same rules to all vehicles = f(standardization, cost to comply) • Apply rules retroactively = f(hardware impacts) • Types of regulations; e.g., dynamic control = f(manned, unmanned, communications time delay, communications link reliability, system reliability, docking/berthing, hardware & software)
5. Parameters to be monitored	• Regulations, zone size, range • Monitor only observable parameters = f(manned, unmanned, base's sensor array) • Monitor telemetry; e.g., safety critical systems data = f(communications link reliability, hardware & software impacts, ground processing time)
6. Duration of each zone's activation	• Impacts of continuous activation • Impacts on manned base
7. Communications and telemetry compatibility	• Standardized parameters • Manned base's antennae coverage = f(zone regulations, range, relative attitude) • Level of conformity = f(technology) • System automation level = f(cost, technology)
8. Tracking compatibility	• Sensor type = f(trajectory, zone regulations) • Standardized parameters = f(trajectory, zone regulations)

* The following notation is utilized to explain the related issues:
- Each issue is shown on the left side of the equation and each factor that influences it is shown on the right side.
- Issue = function of (various factors) = f(factor #1, factor #2, factor #3)

Therefore, the point to be decided is: how many interactions must occur over the program lifetime before the cost of mission-unique efforts exceed the cost of conforming to international standards? One factor is the class of manned vehicle being considered for a control zone. It is possible that cost savings for vehicles with short mission durations (e.g., orbital shuttles) may not match savings for vehicles with longer duration orbital missions (e.g., space stations). Trade studies may show that it is not practical to develop regulations for more than one class of manned base at the present time.

Another area that will influence this decision is vehicle design. It seems reasonable to expect that signatories to an international agreement will incur additional short-term program costs to purchase or develop the flight hardware and software necessary for compliance. For example, hardware items such as sensors are probably required to accomplish any cooperative space activity, and are required regardless of whether or not international standards exist. However, additional costs could occur if the regulations that are eventually developed require sharing of that sensor data between the two vehicles. If the vehicles are already designed or operating, the costs of new designs and retrofitting may be prohibitive. However, it may be possible to design one device that conforms to international standards and procure multiple copies of it. Thus, the nation could realize a long-term savings.

The impact of standards on mission design, mission management parameters, constraint development, etc. should also be considered. Standardization is very desirable in these areas because of the amount of work required to generate mission-unique data products and train ground and flight crews. This process could evolve to a point that a standard set of products is always required to interact with a given vehicle. That would eliminate the need to negotiate new products and data for each rendezvous mission. In a sense, this is similar to air traffic control. A controller in London receives a flight plan with all the necessary information in it before a French airliner reaches his airspace.

Note that the significance of these issues, costs, and savings will vary with the types of regulations that are eventually developed for the zones. For example, if participants agree that direct trajectory monitoring is not necessary, then a radar may no longer be needed onboard the manned base.

2. Classes of vehicles that must comply with control zones

The international community should first consider whether a manned base's control zone should apply to all spacecraft that enter it, or only to those spacecraft that intend to interact with the base. For example, an operational satellite may enter a control zone occasionally, but never plan to interact with the manned base. Establishing zonal compliance for such vehicle classes may prove to be an unreasonable cost impact. This impact is particularly a concern for manned bases operating at high altitudes, such as geosynchronous Earth orbit. Accordingly, the international community must consider the actual motivation for requiring compliance, and determine if any valid workarounds exist. For example, near-term concern for potential collisions with operational satellites might be addressed by arranging for some organization to notify the manned base when a spacecraft is going to enter its vicinity. The manned base could then go to "alert" status for potential collision avoidance maneuvers. Then, over the long-term, the international standards could be expanded to allocate additional zones for unmanned vehicles such as satellites. Manned spacecraft could then be restricted to operate outside of these zones.

Next, consider the vehicles that will interact with the manned base. One means of determining which classes of these vehicles should be subject to compliance is to consider how frequently the interactions are expected to occur. Obviously, the more frequently a spacecraft enters a manned base's control zone, the stronger the case for establishing zonal compliance. Such compliance simplifies interfaces, enables mission standardization, etc., as was discussed in an earlier section. However, this standardization may once again require a trade study to determine how frequently the interactions must occur before the cost of mission-unique development and planning exceeds the cost of standardization.

Another matter that will influence this decision is the controllability of the spacecraft in question. A spacecraft approaching the manned base must provide an adequate margin of safety for the base's crew. The question to be decided is what constitutes an adequate safety margin? Manned vehicles may not be as critical in this regard as other classes of spacecraft because a crewmember is a good monitoring system. He/she can see out the window, adapt to trajectory dispersions, and make rapid decisions during contingencies. Since a crewmember's abilities may provide the necessary safety margin, one could make a case to exempt or limit manned vehicle compliance with such regulations.

The controllability of unmanned vehicles may be less certain. By their nature they may represent an increased threat to a manned base (i.e., there is no one aboard). Teleoperated vehicles present a potential safety hazard during proximity operations and docking because of the communications time delays that slow the pilot's ability to react to dispersions and/or contingency situations. An additional safety concern may occur if the communications link fails during proximity operations. In both of these cases, the incoming vehicle could be relatively uncontrolled and on a collision course with the manned base. Both of these concerns will be discussed at greater length in section 4 of this paper.

Autonomous vehicles may have less controllability concerns than other unmanned vehicles. Once they are tested and certified, concerns about time delays may be reduced because everything is computed and executed onboard. It is still expected that the manned base's crew will want to monitor the incoming vehicle's trajectory and possibly its systems, and precedent exists for such monitoring. For example, Soviet cosmonauts monitor the trajectory of Progress tankers by television during the final phases of their docking operations with the Mir Space Station.

In section 1, it was observed that assigning a control zone to a given manned base might increase its hardware and software requirements. This increase may also be true for the classes of vehicles required to comply with those control zones. The extent of these impacts will be a function of the spacecraft capabilities, the regulations that apply within a control zone, and the size of the zone. However, in general, it can be assumed that spacecraft already planning to interact with the base will experience less impact from compliance than transient vehicles. For example, payloads may already provide safety critical data to the U.S. Space Shuttle, so there may be limited impact if these data must also be provided to a space station. However, transient vehicles may require system changes, with the associated weight, power, and cost penalties, to comply with the zone.

Zonal compliance may also be expected to increase both the amount of planning to be done and the mission management data to be generated. If a vehicle already plans to interact with the manned base, then a significant amount of premission coordination will be conducted regardless of the existence of a control zone. The existence of a zone may require some new data to be generated, but this should be limited because similar types of planning data are probably necessary in either case. However, the manned base may incur additional planning costs if its support personnel are required to generate the plans for interacting vehicle's contingency operations within the control zone. Finally, depending upon the control zone's regulations, training may also be expected to increase. For example, if the manned base's crew is to monitor the interacting vehicle's systems data, then they (and possibly the ground controllers) must be trained to understand it.

These effects on planning and training are magnified if transient vehicles are required to comply with zonal authority. In this case, the additional coordination and regulations may represent a significant planning overhead to these spacecraft, because they may not have planned to generate such mission management parameters.

3. Size of zone allotted to each type of manned base

Safety requirements may be expected to have a significant influence on the size of a manned base's control zone. In short, the zone must be sized to provide adequate time for the flight crew to recognize a problem and respond to it. Some of the many factors that influence this issue are discussed below. Note that in all cases described below, it may be difficult to infer the zone size for an entire class of vehicles from the results derived from one specific example. Therefore, sizing studies should consider several vehicles in the same class when assigning the manned base's zone size.

It has been observed that safety requirements may vary with the class of vehicle interacting with the manned base. Accordingly, vehicle class may be expected to influence zone sizing. For example, it is possible that detailed risk assessments may require monitoring of unmanned vehicles at greater ranges than a manned vehicle, simply because of the controllability concerns discussed earlier. If this proves to be the case, then the zones must be sized to provide adequate sensor and communications coverage for the classes of vehicle in question.

The trajectory followed by the interacting vehicles may also be a factor, depending on the zone regulations that develop. For example, zone regulations may require that the manned base monitor the critical portions of the interacting vehicle's trajectory, such as execution of the intercept maneuver. To do this monitoring, the zone should be sized to include those trajectory phases. This monitoring was one factor that influenced the sizing of Space Station Freedom's command and control zone (i.e., it was sized to allow tracking of the Orbiter for 1/2 orbit prior to the intercept maneuver). Obviously, the trajectory is a function of many things, including trajectory dispersions, crew activity plans, etc. Hence, each of these things will also influence zone sizing.

The relative velocity of the approaching vehicle must be considered. The faster a vehicle approaches the manned base, the larger the zone may need to be in order to allow the same amount of monitoring time. This zone size may be less of a concern for vehicles engaged in rendezvous and docking, as relative velocities are generally well controlled by the trajectory design. However, it may be a serious consideration if the zonal authority extends to transient vehicles such as satellites.

Safety should consider whether or not the manned base is capable of performing evasive maneuvers. The zone should be sized to allow adequate time to react to collision threats. Some of the elements influencing reaction time are the manned base's sensor capabilities and the acceleration capability of the interacting vehicle. A manned base with significant capability to perform evasive maneuvers can reduce the reaction time which might reduce zone sizing. Caution is urged in exercising this means of reducing zone size, however. Cooperative aborts will result in two vehicles maneuvering in close proximity to each other which may raise more safety concerns than it solves; cooperative aborts may be much more complex because the dispersions and failure modes of two vehicles must now be considered. Planning such maneuvers may increase the premission planning, training, and system verification.

Another issue in determining zone size is the operating altitude of the manned base. Currently, large volumes of space exist between various spacecraft in orbit. However, as the on-orbit population increases, the competition for available space can be expected to increase. Such competition is already evident for satellites in geostationary orbit. It therefore seems likely that the international community will wish to limit the size of zones in order to ensure equal access to the orbit resource. Alternatively, such population increases represent increased activity in the vicinity of the manned base, and as such, could be grounds for larger zones, or at least additional zones with different regulations.

While some issues related to hardware, software, planning, and mission management parameters have already been discussed, it is difficult to identify the full range of such issues and to quantify their significance in determining zone sizing. It is probably safe to say that the larger a zone is, the greater its impact on these items. For example, monitoring a large zone may drive additional weight, volume, and power requirements for equipment such as radar aboard the manned base. Large zones may also increase premission planning because more phases of the trajectory must be examined. This increased planning will involve more disciplines to assure zonal compliance. Monitoring more trajectory phases may also require additional mission management parameters to be developed premission and monitored in real time. This additional development will also increase planning costs and require additional training. Alternatively, it may not be possible to increase certain hardware parameters such as power levels. Thus, hardware limitations may feedback into this process and limit zone size and shape.

Finally, proprietary and security concerns may also be expected to influence zone sizing, but they are outside the scope of this paper. For example, it is unknown whether distancing vehicles is an acceptable means of ensuring proprietary and national security.

4. Regulations that apply within control zones

One issue is whether the same regulations should apply to all interacting vehicles, or should they vary by vehicle class. Applying the same set of rules to all interacting vehicles will enhance standardization and may reduce training. However, it may also drive excessive and unnecessary hardware and software development because it seems unlikely that all vehicles will require the same level of monitoring. Thus, standardization in this regard runs the risk of overspecifying the solution at a significant increase in implementation cost.

The community should also consider the worth of retroactively applying the regulations to existing spacecraft or those far along in their design cycle. In some cases there will be only limited conflicts between the regulation and the vehicle's current capability. However, other cases are likely in which the incompatibility and resulting hardware impacts could be more extreme. For example, interfacing to Orbiter avionics in nonstandard ways can be difficult due to space limitations for cabling. Hence, some criteria must be developed to decide when the cost of vehicle modification outweighs the need for compliance. Regulations might be assigned a graduated importance, where those related to safety critical concerns are levied on the existing vehicle and others are addressed by operational workarounds or waivers. However this is resolved, it should be noted that there is at least some precedent for requiring vehicle modifications to meet new safety standards after development is under way. Following the Challenger accident, updated safety requirements were levied on all payloads slated to fly on the Orbiter, regardless of their state of development at the time.

This paper discusses issues rather than specific proposals. Thus, it does not discuss specific regulations. However, there are certain types of regulations that may be common to various zones. Discussion will now address the issues associated with types of regulations that require dynamic control, because they are illustrative of issues that may be encountered when defining the actual regulations and because they are of specific interest to the international community.

The community should consider the necessity for those types of regulations that require dynamic control of specific vehicle classes from a manned base. A regulation of this type might require that the manned base's personnel remotely pilot the interacting vehicle when it is within the control zone. Such regulations may be strongly driven by safety concerns, and are therefore a function of vehicle class, system redundancy, and other factors. There is probably no need or intent in the international community to remotely pilot an interacting manned vehicle. Therefore, consider the issues that influence the applicability of this type regulation to various classes of unmanned vehicles.

One influence could be the existence of a communications time delay between the ground-based pilot and the orbiting spacecraft. This is the case with some teleoperated vehicles, such as the U.S. orbital maneuvering vehicle. Such time delays can make it difficult for the pilot to rapidly respond to changes in relative position and attitude under nominal conditions, much less contingencies such as failed-on thrusters. Thus, teleoperation from the ground could present an increased threat to the manned base and/or a reduced probability of mission success for the vehicle. Lower mission success rates may be unacceptable for operations involving manned vehicles. However, the significance of this issue will also be a function of the minimum translational and rotational rates that the vehicle can consistently maintain during those operations. For example, small minimum rates can be expected to increase the pilot's control over the trajectory and can also reduce the amount of DV inadvertently applied in the event of a failed-on thruster.

Another issue for teleoperated vehicles is the reliability of the communications link with the ground. A link that is unreliable may increase the risk of collision if it fails at critical points in the trajectory (e.g., during docking operations) which could, in turn, drive a need for dynamic control regulations. However, it may be possible to reduce the significance of this issue through vehicle design. For example, safety could potentially be improved if the teleoperated vehicle is designed to perform automated hold or abort maneuvers in the event that the link to the ground fails during critical mission phases. The U.S. is currently developing such a capability for its orbital maneuvering vehicle program[5]. Two items must be noted in this regard. First, the success and acceptability of this type of capability has yet to be proven, particularly in the vicinity of manned vehicles. Secondly, successful automated abort capability will do nothing to improve the spacecraft's ability to complete the mission (i.e., the probability of mission success) in the event of a link loss. That ability will be a function of how quickly the ground recovers control and the orbital mechanics of the problem.

These concerns may not effect autonomous vehicles to the same degree as they do teleoperated vehicles, since they will presumably be designed to rendezvous and dock without nominal ground intervention. Instead, it is the reliability of such autonomous systems that may dictate the need for dynamic control. For example, if the performance of such vehicles is successfully demonstrated, crew safety might be assured by providing an abort command from the manned base. Presumably, the spacecraft could back away and the ground could assess the problem which is similar to the procedure utilized by the Soviet Progress Tanker. It should be noted that it is uncertain what effect such an abort might have on the probability of mission success, as this is undoubtably a function of vehicle capability and the orbital mechanics.

Dynamic control regulations will also be influenced by whether the unmanned vehicle will dock with the manned base, or be berthed by some manipulator mechanism. Depending upon the reach distance of the manipulator, berthing may represent less of a threat to the manned base than docking. Likewise, it will be influenced by whether the manned base is capable of performing evasive maneuvers. Such capability may also reduce the threat of collision and the need for dynamic control regulations.

The hardware and software impacts of implementing dynamic control must also be considered because it seems likely that it will require additional capability on both vehicles. For example, the manned base may require special translational and rotational handcontrollers, displays, navigation and control algorithms, and television. However, some of this equipment may be required regardless, to successfully perform other monitoring functions within the control zone. Next, consider the training and planning impacts. Dynamic control regulations will increase crew training requirements for the manned base. In addition, maintaining these crew skills may be difficult if the designated pilot is on-orbit for a long period of time prior to piloting the unmanned vehicle. Hence, onboard "refresher" training may also be required. As was noted earlier, dynamic control could also require the manned base or its ground crew to perform contingency trajectory planning. Note that these concerns may be reduced for autonomous vehicles if an abort command is utilized in place of dynamic control.

5. Parameters to be monitored

The community must decide what types of parameters must be monitored in order to satisfy safety and mission success criteria for international missions. These parameters will be partly a function of the size and regulations established for a given zone. As was stated earlier, large zones may envelop more phases of the trajectory than small ones which, in turn, may increase the number of mission management parameters to be monitored. In addition, the type of data to be monitored is probably a function of the range to the manned base. Some of the parameters to be monitored at long ranges [approximately 50 km (27 n.mi.)] may include relative position, relative velocity, and safety critical system status. At closer ranges, the manned base might also require relative attitude and direct visual sighting. Some of the issues associated with deciding these parameters are discussed below.

First, will the manned base only monitor those parameters that it can observe with its own sensors, or will it require telemetered parameters (e.g., safety critical systems data)? Assuming that there is direct voice contact between the two spacecraft, this issue may not be critical for manned vehicles interacting with the manned base. They have a crew onboard to monitor most of the same parameters that would concern the crew of the manned base. Therefore, consider the issue in terms of unmanned vehicles.

Some parameters can be obtained through either observation or telemetry. For example, relative position can be determined by the manned base's relative sensors or by telemetering data from the interacting vehicle's relative sensors to the manned base. (Note that the interacting vehicles may be required to carry some type of sensing aid, such as corner reflectors or a radar transponder to accomplish this.) In such a case, the decision of which source of data to use may be a function of the sensor array onboard the manned base. If the manned base has no relative sensor and if this type of data is critical, then telemetry may be the only solution.

Other types of data that the manned base might wish to monitor may only be obtainable through the interacting vehicle's telemetry or from the vehicle's ground control center. Safety critical systems data are a probable example. Thus, for the sake of illustrating some of the factors that influence the issue at hand, the remainder of this section discusses the need for safety critical systems data to be monitored onboard the manned base.

The need to monitor telemetry onboard the manned base may once again be a function of the communications link. If both the interacting vehicle and the manned base have highly reliable links to their ground control centers, then it may not be necessary for the manned vehicle to monitor telemetry. However, the international community will need to define what constitutes adequate reliability. In addition, it is possible that a direct communications link may need to be established between the manned base and the unmanned vehicle's control center.

If the communications links are unreliable, then lack of monitoring capability on the manned base may reduce the probability of mission success. Certain parameters must be monitored by either the ground or the manned base before a vehicle will be allowed to approach the base. It seems unlikely that an unmanned vehicle would be permitted to continue its approach if these parameters were not being monitored. The resulting abort may be difficult to recover from due to orbital mechanics, vehicle power limits, etc. In addition, contingency planning for such an abort may require additional premission and real-time planning. However, it may also be possible to improve communications redundancy by planning critical mission phases to occur over ground tracking sites. Unfortunately, this will add yet another constraint for rendezvous mission planners to contend with.

In either case, there will be hardware, software, and training impacts to be addressed. For example, if the manned vehicle does monitor the interacting vehicle's telemetry, it will need communications hardware and software, special displays, crew training to interpret the displays, etc. Without the telemetry link, additional control center links (to the ground tracking sites, etc.) and processing software may be required. In addition, lack of telemetry monitoring capability may require that the unmanned vehicle be capable of performing automated aborts to protect against loss of telemetry downlink.

Finally, it should be noted that significant time delays in processing the telemetry data on the ground may drive a need for onboard monitoring even if there is a reliable ground link. This need will be a function of the magnitude of the delay and the particular parameter in question. Safety critical parameters that can exceed their safety limits very quickly may still require onboard monitoring in order to allow the crew time to react.

6. Duration of each zone's activation

The concept of assigning control zones to manned spacecraft in orbit is relatively new and may be expected to impact both the manned base and the vehicles that interact with it. It is therefore prudent to consider whether such zones should be active continuously, 24 hours per day and 365 days each year, or whether they should be active only part of that time. There appear to be two main issues in deciding this.

First, what are the impacts on other spacecraft of having the zones continuously active? This impact is a function of many variables, and many of the technical issues have been discussed elsewhere in this paper. In summary, it seems reasonable to state that the more inclusive and restrictive zonal authority is, the more desirable it may be to limit the times during which the zone is active. That is, a large zone that requires all vehicle classes to comply with a very strict set of rules will be more of an impact to the international community than one which is not as inclusive.

Second, what are the impacts to the manned base and its mission if the zones do not apply continuously? One could make a case that the sheer value of space station-class vehicles may be such that the sponsoring nations want to maintain some authority over distances of closest approach and other factors. For example, the orbiting elements of Space Station Freedom may cost on the order of $6 billion, including the cost of their engineering development. Thus, it may be better to develop different regulations for noncritical times than to eliminate or deactivate zones. In addition, reducing the duration of zone activation may present proprietary or security concerns that are beyond the scope of this paper.

It is difficult to identify which of the hardware, software, and mission management parameter impacts are most significant in this context without having resolved some of the other issues discussed in this paper. In addition, there are undoubtably other factors that will influence this decision. Therefore, further study is required before this issue can be resolved.

7. Communications and telemetry compatibility

Communications and telemetry will be necessary to execute and monitor operations within a control zone. The data to be transmitted during future international missions may include voice, television, and data transmission. The first issue to be considered is the development of a set of communications standards for use with control zones. Such standards are necessary for two reasons: (1) to assure the establishment of a communications link and (2) to provide uniform and consistent information transfer and exchange.

Without standards for certain basic parameters, the communications link cannot be assured and many operations and safety considerations could be jeopardized. For example, the international community should agree upon the radio frequency of operations for various links (including space-to-space and space-to-ground links). Once the frequencies are allocated, their use should be regulated to avoid radio frequency interference. It should be noted that these frequencies can differ for various links. Another example of a communications parameter that should be standardized is polarization. Communications links can be implemented through linear, circular, or elliptical polarization strategies. Accordingly, coordination is necessary to ensure that the polarization of the receiving antenna matches that of the incoming wave. Similar design considerations apply to each communication link and system parameter, indicating the need for standardization to support cooperative international operations. Other such parameters include data rates, link margins, signal-to-noise ratio, radio frequency interference, modulation and demodulation protocols, and operational modes (simplex, duplex, and multiplex). Discussions leading to communications protocols for assured links should lead to an acceptable and cost-effective communications and telemetry system design to be utilized by the international community.

Standardization should also extend to parameters that affect the processing of information once the communications link is established. These parameters include carrier frequencies, data formats, and coding/decoding schemes. For example, any telemetry data received must be decoded before the data can be utilized by the system monitoring or command processing software onboard the vehicle. In the past, issues related to information exchange have been addressed by the International Telecommunications Union and the Consultive Committee for Space Data Systems. At the present time, information standards such as the Consultive Committee are emerging worldwide and these may influence the development of international standards for control zones.

The matter of adequate antennae coverage required onboard the manned base should also be addressed. Depending on the zonal regulations, coverage requirements for communications and telemetry may be a function of range (see section 5). Ensuring communications coverage at very close ranges can be more difficult than longer ranges, because the relative attitude of the vehicles becomes a more dominant factor. Omnidirectional antennae coverage could be utilized to assure proper coverage at short ranges; however, it seems impractical to implement this for an entire control zone. Various solutions can be envisioned. For example, a single control zone could be divided into several communications regions. One region might include the space within a few hundred meters of the manned base and the other might extend from this near region to a few tens of kilometers. Then, omnidirectional antennae coverage could be specified for the innermost region without severe impacts to the communications and telemetry system design. Without such a strategy, the interacting vehicle may be constrained to approach within a specific cone or region in order to fully communicate with the manned base. This approach in turn, could result in undesirable restrictions on the interacting vehicle's trajectory. Also, if several approach regions are implemented, each link may need to utilize a separate antenna to provide adequate coverage. These considerations are important for uniformity of communications and telemetry system designs for the international community.

Another issue is determining at what level communications conformity should be required. The communications hardware implementations utilized by various nations have evolved differently depending on their needs and technological advancement. One obvious step to communications and telemetry conformity would be to specify hardware designs and subsystems. However, this approach could lead to technology transfer issues and concerns. On the other hand, specification of hardware <u>function</u> and the resultant overall communications and telemetry <u>performance</u> would alleviate these concerns. For example, instead of specifying a distributed array antenna with an agile beam, one can specify the spatial and spectral coverage, and gain of the antenna and allow the implementing nation to decide an antenna configuration and type.

The international community should also determine what level of communications and telemetry system automation is required. Such automation allows fault detection and recovery and selection of the appropriate assets (e.g., receivers, antennae, transmitters) for various links. Coordination is necessary to ensure that a vehicle with automated features (e.g., selection of gains) implements the capabilities necessary to interact with vehicles that perform such functions manually. Note that this type of automated operational capability is currently being developed for possible implementation in Space Station

Freedom's communication and tracking subsystem. However, increased automation could increase the cost of hardware and software system development. The software resident on the communications and telemetry system should also be considered for mutual acceptance. Such standardization would be beneficial in the reduction of estimated program costs and could assist in contingency situations. It may also present technology transfer questions that are beyond the scope of this paper.

8. Tracking compatibility

Two issues should be addressed in order to achieve tracking compatibility. They are discussed together because they are functions of many of the same factors. First, the types of sensors that will be utilized within a control zone should be agreed upon. Second, a set of standards should be developed for the system operation parameters and hardware specifications. Table 2 is an example of specifications that have been proposed for the sensors to be utilized in U.S. space operations[8]. Other specifications could include the bands of operation, polarization, look angles, coverage, data rates, field-of-view, and data formats.

Table 2 Proposed specifications for sensors used in U.S. space operations[7]

System reliability	>0.9999
System weight	<35 kg (77 lb)
System power	<150 watts
Range resolution	≤0.5 cm (.2 in)(0 - 1 km) ≤1% R (1 - 100 km)
Range rate resolution	≤0.3 cm/sec (0 - 1 km) ≤0.002 R1/3 (1 - 100 km)
Bearing resolution	≤2 deg/R1/3 (0 - 1 km) ≤0.05 deg (1 - 100 km)
Bearing rate resolution	≤0.1/R 1/2 deg/sec (0 - 1 km) ≤0.002 deg/sec (1 - 100 km)
Sensor sample rate	≥10 samples per sec

The specific accuracies and types of data required to meet nominal and contingency trajectory and control zones requirements will influence each of these issues. Consider the choice of sensors. The Apollo Soyuz ranging and tracking equipment included optical, television, radar, docking targets, and lights. Future missions envision the use of these and other sensors. For example, infrared systems may be necessary for vehicle detection in the absence of natural or artificial light (e.g., during the dark portions of an orbit). Laser vision and laser radars are useful for determining position, velocity, and attitude with extremely high accuracy. This equipment may be required to support some docking operations. Hardware standards are similarly affected by the operational requirements. Therefore, a concerted effort should be made to standardize ranging and tracking requirements for international operations. These standardized requirements would simplify the definition of the other tracking issues.

Finally, technological advances such as automatic operation and fault tolerance will eventually become available and be implemented. These hardware implementations can result in technology transfer issues and concerns that must be addressed.

Conclusions

A control zones concept will provide a consistent foundation and an integrated framework for the development and conduct of international space operations. Initially, it can be utilized to coordinate various types of unmanned activities. The consistent framework provided by such a strategy will also support early definition of requirements for international missions. For example, the concept originally adopted for Space Station Freedom has assisted requirements definition for Europe's Man-Tended Free-Flyer.

This paper identified a broad range of issues to be considered in developing a control zones strategy. At this point it is prudent to mention some additional areas for future consideration. First, considering the high cost of activities in space, the international community may wish to define what parameters constitute grounds for aborting a mission (when they exceed their nominal ranges). Second, the issues discussed herein only considered contingencies for the interacting vehicles. The community should also evaluate the need for control zones when the manned base suffers a failure. Specifically, do zones offer any benefits then and how would the regulations change as a result? Third, this paper focussed primarily on the orbiting spacecraft themselves. However, some of the decisions to be made when establishing control zones may be influenced by impacts to existing ground facilities. Factors such as control center interfaces with other facilities must ultimately be considered.

Next, the community should examine the benefits of developing a set of standards for systems redundancy. For example, unmanned vehicles that were designed to operate with only other unmanned vehicles may not be redundant enough to satisfy safety requirements for docking with manned spacecraft. Under these circumstances, the requirements could be met by upgrading the unmanned vehicle or, conceivably, by adding the redundancy to the manned base or the ground. The availability of such standards could reduce future retrofitting of systems by specifying whose responsibility it is to provide adequate redundancy early in a program's design cycle. Finally, how could control zones be modified to support lunar bases and Mars missions? For example, it might be beneficial to assign a parking orbit zone as a holding orbit for freighters carrying lunar materials to Earth orbit.

References

1. B.A. Nader and A.L. DuPont, "Space Station Operations: Operational Control Zones," JSC-20235. Mission Planning and Analysis Division/JSC, January 1985.

2. "Space Station Lexicon," JSC-31012. Space Station Projects Office/JSC, January 1987.

3. "Space Station Program Definitions and Requirements, Space Station Systems Requirements," Revision H, June 30, 1989, paragraphs 3.1.18.9, 3.2.7.1.5, and 4.1.3.

4. "ASTP Flight Mission Rules," JSC-09450. Flight Control Division, Flight Operations Directorate/JSC, November 14, 1974.

5. "Software Requirements Specification for the Orbital Maneuvering Vehicle Operational Flight Program," FSCM No. 11982. TRW Space and Defense Sector, Redondo Beach, California 90278. September 2, 1988.

7. DiBattista, et al., "Autonomous Systems Control for Solar Systems Exploration Spacecraft," The Second AIAA/JPL International Conference on Solar System Exploration, California

Institute of Technology, Pasadena, California, August 22-24, 1989.

8. "Apollo Soyuz Test Project, Summary Science Report," Vol. I, NASA SP-412, Scientific and Technical Information Office, NASA, Washington, D.C., 1977.

N91-20669

A perception and manipulation system for collecting rock samples[1]

T. Choi, H. Delingette, M. DeLuise, Y. Hsin, M. Hebert, K. Ikeuchi

The Robotics Institute,
Carnegie Mellon University,
Pittsburgh Pa15213

Abstract

An important goal of a planetary exploration mission is to collect and analyze surface samples. As part of the CMU Ambler project, we are investigating techniques for collecting samples using a robot arm and a range sensor. The aim of this work is to make the sample collection operation fully autonomous. We describe in this paper the components of the experimental system that we have developed, including a perception module that extracts objects of interest from range images and produces models of their shapes, and a manipulation module that enables the system to pick up the objects identified by the perception module. We have tested the system on a small testbed using natural terrain.

1 Introduction

One of the most important goals of a planetary exploration mission is to collect and analyze terrain samples. As part of the CMU Ambler project [2], we are investigating techniques for autonomously collecting samples. We have developed a system that is able to collect small rocks using computer vision and planning. Our goal is to eventually integrate the system to the Ambler system, a six-legged autonomous robot for planetary exploration.

We have developed a rock sampling system that includes: a robot arm, a range finder, and a small terrain mock-up that contains sand and small rocks. The goal of the rock sampling system is to identify, locate, and pickup rocks from the terrain. The control flow of the rock sampling system is shown in Figure 2: First an range image of the scene is taken and features are extracted from the image (Section 2). The features are surface features such as surface discontinuities that are used to extract the object boundaries. Then the contours of the objects in the scene are extracted. Since, we are dealing with natural environments, we make very weak assumptions on the possible shapes of the objects and on the distribution of the features in the image. To handle those constraints, we have developed a new shape extraction algorithm (Section 3.1) based on the concept of deformable contours. The set of points enclosed by the contour of an object is approximated by a superquadric surface (Section 3.2). In some cases the object representation using superquadrics may not be sufficient. An algorithm based on deformable surfaces can extract directly a surface representation of an object using the image features without relying on superquadric fitting (Section 4). Finally, the parameters of the surface that approximate each object (superquadric or deformable surface) are used to grasp it using a clam-shell gripper (Section 6). The algorithms for object extraction assume that there is an initial guess of the positions of the objects in the image. We present an algorithm for selecting the object location hypothesis automatically in Section 5.

[1]This research was sponsored by NASA under Grant NAGW 1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the US Government.

2 Image acquisition and feature extraction

In order to manipulate objects, we need an accurate description of their shape. This implies that we need to use a sensor that can sense the 3-D surfaces observed in a scene. Therefore, the only possibility is to use a sensor that measures range data. Many range sensing techniques are available [3]. The range sensor that we are currently using is an active sensor that consists of a projector equipped with a computer-controlled LCD screen and a camera [14]. The projector illuminates the scene through the LCD screen. As several illuminations patterns are projected, the corresponding images of the scene are collected by the camera. The range to each point in the scene is recovered from the shape of the projected patterns. The output of the sensor is set of four 256×256 images: an intensity image and three images, X, Y, and Z that contain the three coordinates of the three spatial coordinates of each pixel. The coordinates are with respect to a fixed reference frame defined at calibration time. The spatial and range resolution of this sensor is appropriate for this application in which we need high-resolution measurements at close range. We currently use the intensity image for only display purposes although it could also be used in the object extraction algorithms [12].

Figures 3 and 4 show the images of two scenes. The upper left image is the intensity image, the other three images are the coordinate images. The coordinate images are coded on 16 bits and displayed on 8 bits which accounts for the periodic effect in those images. Figure 5 shows a 3-D display of the data from Figure 3.

Once an image is acquired, the next step is to extract features of the terrain that can help extract the objects of interest in the environment. Many different types of features can be extracted from range data [4] ranging from planar facets to local extrema of the principal curvatures [5]. However, most of those techniques do not apply to this problem mainly because we are working in an unconstrained natural environment which rules out all the feature types, *e.g.* planar or quadratic patches, that assume a known geometric structure of the environment. Furthermore, it is our experience that the standard techniques based on curvature analysis perform well only when the data is very accurate and well distributed. We have chosen an approach in which we detect local features that are relatively insensitive to noise. We do not force the features to provide a complete description of the terrain, in particular we do not expect those features to connect to each other to form the boundaries of the objects in the scene. Instead, we want each feature to give partial evidence of the presence of an object in its vicinity. Grouping the detected features into objects is the job of the segmentation algorithms introduced in the next Section.

Three types of features are extracted:

- Range shadows: Objects produce shadows in the range images, which are areas of the scene that are illuminated by the projector but that are not visible from the camera because they are occluded by an object's surface. This phenomenon occurs with any sensor that uses a triangulation technique. Range shadows are therefore a important cue for the extraction of objects. Extracting the range shadows does not require any image processing since they are identified by the sensor itself.
- Surface discontinuities: A surface discontinuity is a large variation of range between neighboring points in the image. Such discontinuities occur mostly in the vicinity of the occluding edge of an object. Surface discontinuities are detected by applying an edge detector to the image of the range values, $r = \sqrt{x^2 + y^2 + z^2}$. The final edges are obtained by thresholding the resulting edge magnitude. The threshold is computed from the distribution of the edge magnitudes in a large window centered at each image pixel. The reason for using a variable threshold is that the range r varies more rapidly as points are measured further from the sensor. Spurious edges would be detected if a fixed threshold were used.
- Surface normal discontinuities: Surface normal discontinuities occur when two surfaces intersect as is the case when an object is resting on top of the terrain. The normal discontinuities are detected by first computing the unit surface normal n at each point and by finding the low values of the dot products $n_1 \cdot n_2$ of the surface normals at adjacent pixels. The three coordinate images must be smoothed first since the surface normal computation is quite sensitive to noise in the data. Further smoothing is applied to the surface normals.

The image pixels that are labeled as one of the three feature types are grouped into connected regions. The set of feature regions is the input to the segmentation algorithms. Figure 6 shows the features computed from the image of Figure 3. The features are shown as shaded regions. As expected, the features are concentrated around the objects although some are detected on the underlying terrain and no group of features form a closed object boundary.

3 Object extraction: deformable contours and superquadrics

The features give an indication of where the boundaries of the objects may be located in the scene. However, the raw features are not sufficient for reliably extracting the objects from the scene because the objects may be small or partially buried in the terrain. Therefore, we cannot use a simple region extraction that would assume that the features are grouped into closed boundaries. Instead, we used the concept of deformable contours and deformable surfaces. The idea is that a contour that is attracted by 2-D forces generated by the detected features and by the data points measured on the terrain is iteratively deformed until the forces applied to it are in equilibrium. A smoothness constraint is added to the forces so that the contour or the surface does not have sharp discontinuities of orientation or curvature. The final product is a smooth contour that approximates the shape of an object that is partially enclosed by features. The advantage of this approach is that object descriptions can be extracted from the image even if only few scattered features are observed. This is in sharp contrast with other vision problems such as model-based object recognition in which an accurate model of the objects is known apriori. We do not make any assumption on the shape of the objects other than a maximum and minimum object size, and we do not make any assumption on the configuration of the features.

This approach is inspired from Witkin's "snakes" [11] and from Terzopoulos' symmetry-seeking surfaces [16]. We describe in detail the deformable contours algorithm in the next Section.The algorithm assumes that one point that lies inside the object is initially selected. The actual selection of this starting point is the object of Section 5. We assume for now that this point is available. Once a contour is extracted, a three-dimensional model of the corresponding set of points must be built. We use superquadrics to represent the object models (Section 3.2).

3.1 Deformable contours

A deformable contour is a contour in a range image that is subject to forces that change its shape over time. The contour reaches a stable shape when all the forces are in equilibrium. The points that are inside the region enclosed by the final contour are used to described the shape of the object. The algorithm used to derive a shape representation from the region is described in Section 3.2.

We represent a contour by an ordered set of pixel (r_i, c_i) where r_i is the row coordinate in the image, and c_i is the column coordinate. A 3-vector p_i, that is the position of the scene point measured at pixel (r_i, c_i), is associated with each pixel. In addition, the normal to the contour n_i is defined at each p_i. The n_i's are two-dimensional vectors expressed in image coordinates. Furthermore, n_i is always oriented from the inside to the outside of the contour. It is always possible to define such an orientation since the contour is guaranteed to be closed without self-intersections. Each p_i is subject to a set of forces. Each force is a signed scalar that indicates in which direction p_i is attracted. A positive force indicates that p_i is attracted toward the outside of the contour in the direction of the nearest feature. The algorithm is designed in such a way that the contour can only grow outward.

Each pixel of the contour is subject to two types of forces 7(a). The external forces are exerted by entities that are not part of the contour such as features. The internal forces depend on the contour itself and are independent of the data. Internal forces are typically used to force the contour to be as smooth as possible.

The first external force is generated by the features. It is an attractive force defined at each point p by:

$$F_{\text{feature}} = \sigma_{\text{feature}} \left(\frac{\|p - \mathcal{F}(p)\|}{R_{max}} \right) \tag{1}$$

where $\mathcal{F}(p)$ is the point of the image features that is the closest to p, σ_{feature} is a function that relates the force to the distance between contour point and feature (Figure 7(b)), and R_{max} is the maximum expected object size. The closest point $\mathcal{F}(p)$ is calculated by searching the feature points along 16 directions around the contour normal. Since this is a potentially expensive operation, we use several constraints to limit the search: First, the features that are too far from the contour point are not considered. Second, we use the fact that the order in which features appear around an object is defined by the geometry of the sensor and can be computed beforehand thus eliminating features that cannot be part of the current object.

The second type of external force is generated by the starting point. Its purpose is to prevent the contour from "overgrowing" by generating an attractive force towards the center point. The force is defined by:

$$F_{\text{center}} = \sigma_{\text{center}} \left(\frac{\|p - p_0\|}{R_{max}} \right) \tag{2}$$

where R_{max} is defined as before, p_0 is the starting point, and σ_{center} is the attraction function (Figure 7(c)).

The purpose of the internal force is to guarantee that the contour is reasonably smooth. The idea is to make the shape of the contour close to an ellipse. To do that we approximate the contour by an ellipse $\mathcal{E}$ of equation $(p - p_{\mathcal{E}})^t A(p - p_{\mathcal{E}}) = 1$, where $p_{\mathcal{E}}$ is the center of the ellipse, and A is a 2×2 symmetrical matrix. The distance between p and $\mathcal{E}$ is defined by:

$$D(p, \mathcal{E}) = \frac{|(p - p_{\mathcal{E}})^t A(p - p_{\mathcal{E}}) - 1|}{2||A(p - p_{\mathcal{E}})||} \qquad (3)$$

$D(p, \mathcal{E})$ is an approximation of the Euclidian distance between p and $\mathcal{E}$. The internal force is defined by:

$$F_{\text{internal}} = \sigma_{\text{internal}}\left(\frac{D(p, \mathcal{E})}{K}\right) \qquad (4)$$

where σ_{internal} is the attraction function (Figure 7(d)), and K is a constant that controls how far from an ellipse the contour is allowed to be. In practice $K = 0.4$.

The contour deforms itself iteratively. At each iteration, the internal and external forces are computed at each point. Each point is moved according to the resulting force, The complete algorithm follows two steps:

1. Initialize: The initial contour is a small contour centered at the starting point.
2. Iterate: The following steps are iterated until the contour does not deform itself significantly.
   - At each point p_i compute the sum of the forces: $F = F_{\text{feature}} + F_{\text{center}} + F_{\text{internal}}$.
   - p_i is moved by one pixel in the direction of the nearest feature point $\mathcal{F}(p_i)$ if $F > 0$.
   - Resample the contour after all the contour points have been moved according to the forces.
   - Estimate the best-fit ellipse $\mathcal{E}$.

Provided that there is a reasonable starting point, this algorithm produces object contours that are quite good approximations of the true object contour even if the features are very sparse. Figure 8 shows the regions that have been found for each object in the scene using the feature of Figure 6. The starting points were selected automatically using the algorithm of Section 5.

3.2 Superquadrics

Once regions corresponding to objects have been segmented out using the deformable contour algorithm, the corresponding set of 3-D points must be grouped into a surface representation. The resulting object models are used to compute grasp position and manipulator motion.

Although one could use the set of 3-D points computed by the segmentation directly, we use superquadrics to represent the objects. Superquadrics are generalizations of quadric surfaces [1] that can represent a wide variety of shape. Using superquadrics present several advantages: First, it is a compact representation that allows us to represent a wide range of surfaces using a small set of parameters. Second, it provides a global representation of an object whose surface is only partially visible. Lastly, the parameters of a superquadric surface are easily recovered from the coordinates of a set of points.

Superquadrics are described by an implicit equation $F(x, y, z) = 1$, where:

$$F(x, y, z) = \left(\left(\left(\frac{X}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{Y}{a_2}\right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{Z}{a_3}\right)^{\frac{2}{\epsilon_1}} \right)^{\epsilon_1} \qquad (5)$$

and where (X, Y, Z) are the coordinates of (x, y, z) after transformation by a rigid transformation that defines the position and orientation of the superquadric, and a_1, a_2, and a_3 are the sizes of the superquadric along the three directions. Superquadrics can represent a variety of shapes form cubes to ellipsoids by varying the two "roundness" parameters ε_1 and ε_2. Other parameters such as bending and tapering can be included in the equation. To recover the superquadric from a set of points, we use the Levenberg-Marquart minimization approach suggested by Solina [1]. In this approach, the input set of points is first approximated by an ellipsoid which constitutes the starting point of the minimization, then an error function of the form:

$$E = \sum_{(x,y,z)\text{ data point}} (F(x, y, z) - 1)^2 \qquad (6)$$

is minimized with respect to the parameters of the superquadric. This approach works well in our case in which a dense set of points is measured on a portion of the surface (see [13] or [9]) for other superquadric fitting techniques).

Figure 9 shows the superquadric models of the objects found in Figure 8. The models are displayed as wireframes superimposed on the intensity image.

4 Object extraction: deformable surfaces

Deformable contours extract the objects by using essentially the geometry of the scene in the image plane. The result is a region in the image that has to be processed further to yield a complete description of the object. A more direct, although more costly, approach would be to directly find the closed surface that best approximates the data, that is the 3-D points measured on the terrain and the detected features. This leads to the idea of deformable surfaces which are smooth closed surfaces that are subject to forces from the terrain and the features. As with the deformable contours, the surface deforms itself until it closely fits the observed shape. The advantage is that the resulting closed surface should provide all the information needed to pick up the object. As in the case of deformable contours, the algorithm assumes that an initial point is selected inside each object.

The algorithm operates on discrete data, images and discrete features. However, for the sake of clarity it is best to think first of the case of a *continuous* deformable surface that is subject to forces and deforms itself over time. It can be shown that such a surface would reach a stable equilibrium when the Lagrangian of the system of forces reaches a minimum according to the principle of least action [8]. A similar application of the principle can be found in [17]. The Lagrangian is defined by: $L = T - U$ where T is the integral of the kinetic energy over time and U is the integral of the potential energy. If the surface is parametrized as $x = x(\eta, \omega, t)$, $y = y(\eta, \omega, t)$, $z = z(\eta, \omega, t)$, where t is the time, and (η, ω) are the parameters of the surface, then the problem is to find the function that minimizes L. This is a variational problem that can be solved by applying Euler's equation. To simplify the notations, we will denote the points of the surface by $r(\eta, \omega, t)$, r being the 3-vector (x, y, z), and we will denote the partial derivatives by using subscripts (*e.g.* $r_\omega = \frac{\partial r}{\partial \omega}$). Furthermore, we assume that the parameters η and ω vary between 0 and 1.

The term T depend only on the kinetic energy and can be written as:

$$T = \int_0^{t_0} \int_0^1 \int_0^1 \mu ||r_t||^2 d\omega d\eta dt \qquad (7)$$

where μ is a weighting factor that characterizes the inertia of the surface.

In our case, the surface should be deformed so that the following constraints are satisfied: The surface should be smooth, the surface should be as close as possible to the surrounding features, and the surface should be close to the points measured on the terrain. To satisfy those constraints, the potential energy term U is decomposed into three components:

$$U = U_{\text{smoothness}} + U_{\text{features}} + U_{\text{terrain}} \tag{8}$$

The term $U_{\text{smoothness}}$ encapsulates the constraint that the surface should be smooth and continuous. Formally it is defined by:

$$U_{\text{smoothness}} = \int_0^{t_0}\int_0^1\int_0^1 \alpha_1 \left(\|r_\omega\|^2 + \|r_\eta\|^2\right) + \alpha_2 \left(\|r_{\omega\omega}\|^2 + \|r_{\eta\eta}\|^2 + 2\|r_{\omega\eta}\|^2\right) d\omega d\eta dt \tag{9}$$

The weights α_1 and α_2 control how much importance is given to the smoothness constraint. The surface can have any arbitrary shape if they are equal to zero, on the other hand contributions from the features and the terrain are ignored if they are very large.

The term U_{features} implements the constraint that the surface should be as close as possible to the surrounding features. In order to define it, we first have to define the distance between a point and a feature. In order to do that, we represent each feature $\mathcal{F}$ by the 3-D polygonal approximation of its skeleton which is a set of 3-D line segments. The distance between a point r on the deformable surface and a feature $\mathcal{F}$ is the distance between r and its projection on the set on line segments that describes $\mathcal{F}$. We denote the projection by $r(\mathcal{F})$. Strictly speaking, we should compute the distance between r and all the points of $\mathcal{F}$. Since this is too demanding computationally, we use the polygonal approximation which allows us to compute the projection directly. With this definition of $r(\mathcal{F})$, we define:

$$U_{\text{features}} = K\int_0^{t_0}\int_0^1\int_0^1 \sum_{\mathcal{F}} S(r,t,\mathcal{F})\|r - r(\mathcal{F})\|^2 d\omega d\eta dt \tag{10}$$

where the sum is taken over all the features $\mathcal{F}$, and where K is a weighting factor. If we think of a set of springs linking each point of the surface to each feature, K would be the stiffness of the springs. The attraction exerted by the features is basically proportional to the squared distance between the point and the feature $\|r - r(\mathcal{F})\|^2$. An attenuation factor $S(r,t,\mathcal{F})$ is added to avoid one undesirable effect of the pure spring model: points that are very far from the features are always subject to very strong forces. What we would like instead is to have a strong attraction to all the points to initiate the deformation and to have the strength of the attraction decrease over time. For a given distance $\|r - r(\mathcal{F})\|$, this is equivalent to vary the stiffness of the spring as a function of time. Furthermore, it is undesirable for the features to apply an arbitrary large force to the points that are far away. We need a cutoff distance over which points are not attracted. We define the correction factor by:

$$S(r,t,\mathcal{F}) = \sigma\left(1 - \frac{t_0}{t_0 - t}\frac{\|r - r(\mathcal{F})\|^2}{r_0^2}\right) \tag{11}$$

where the function σ varies from 0 at $-\infty$ to 1 at $+\infty$ (Figure 10). σ implements the idea of a cutoff distance: points that are too far away from the feature are not attracted. The cutoff distance is given by the normalizing term r_0. In addition to the cutoff distance, for a given distance $\|r - r(\mathcal{F})\|$, the term $\frac{t_0}{t_0 - t}$ makes the stiffness of the spring vary over time: The spring is strong a $t = 0$ and weakens over time until it eventually disappears at $t = t_0$ at which time only the smoothness constraint and the attraction from the terrain are taken into account.

Another way to look at equation is to consider the term $K\|r - r(\mathcal{F})\|^2$ as a fitting term in that it forces the surface to be as close as possible to the features, and to consider the term $S(r,t,\mathcal{F})$ as a segmentation term in that it takes into account only the group of features that is close to the starting point. The last term of the potential U_{terrain} reflects the attraction between the surface and the terrain. It is defined as:

$$U_{\text{terrain}} = \beta\int_0^{t_0}\int_0^1\int_0^1 \frac{1}{\|r - r(\mathcal{T})\|} d\omega d\eta dt \tag{12}$$

where $r(\mathcal{T})$ is the data point that is closest to the surface point r. Since the term inside the integral would become arbitrarily large as the surface moves closer to the data, we introduce a cutoff distance D at which the potential stops increasing. The potential is therefore redefined as:

$$\begin{cases} \frac{1}{\|r - r(\mathcal{T})\|} & \text{if } \|r - r(\mathcal{T})\| \geq D \\ \frac{1}{D} & \text{if } \|r - r(\mathcal{T})\| \leq D \end{cases} \tag{13}$$

This potential implements a gravity force that increases as points move closer. This has the effect that the feature term is dominant initially when the surface is far from the observed terrain while the terrain becomes dominant as the surface moves closer to the terrain. Notice that strictly speaking we should take into account the contributions from *all* the data points in the computation of the force applied to a single surface point. Since this is computationally untractable we limit ourselves to the closest data point.

We now have a definition of the function L given a set of features and a set of points measured on the terrain. The problem is now to find the surface $r(\eta,\omega,t)$ that minimizes L. The solution is found by straightforward application of Euler's equation. We obtain the differential equation:

$$\begin{aligned} \mu r_{tt} = \frac{1}{2}\beta P(r) + \alpha_1(r_{\omega\omega} + r_{\eta\eta}) - \\ \alpha_2(r_{\omega\omega\omega\omega} + 2r_{\omega\omega\eta\eta} + r_{\eta\eta\eta\eta}) + \\ KS(r,t,\mathcal{F})(r - r(\mathcal{F})) + KS_1(r,t,\mathcal{F})\|r - r(\mathcal{F})\|^2 \end{aligned} \tag{14}$$

where S_1 is computed from the first derivative of σ: $S_1(r,t,\mathcal{F}) = -\sigma'(1 - \frac{t_0}{t_0-t}\frac{\|r-r(\mathcal{F})\|^2}{r_0^2})\frac{t_0}{t_0-t}\frac{r-r(\mathcal{F})}{r_0^2}$, and P is the gradient of the potential due to the terrain attraction, that is the integrand of U_{terrain}: $P(r) = \frac{r-r(\mathcal{T})}{\|r-r(\mathcal{T})\|^3}$.

Applying Euler's equation solves the problem in the case of a *continuous* surface subject to the attraction of the features and the terrain and to a smoothness constraint. To actually compute a solution to the resulting differential equation, we need to construct a discrete approximation of both the surface, that is a discretization of the parameter space (η,ω) and of the time t. Let us consider first the case of the parameter space. Using a straightforward discretization of η and ω in regular intervals of $[0,1]$ would lead to serious problems at the edges of the parameter space just like sampling a sphere along the meridians and parallels leads to problems at the two poles. Since it is not desirable to be forced to handle special cases in the discretization, we would like to use a representation of the parameter space that is as uniform as possible. To do that, we first create a unit sphere that is tesselated using the icosahedron decomposition [6, 7], each point M_i of the tesselation is parametrized by its spherical coordinates (η_i,ω_i) and is a sample point on the surface. The tesselation of the sphere has the property that it is very uniform and that it does not exhibit any poles. With this representation the integrals become sums over the sample points, for example the integral with respect to (η,ω) in U_{terrain} becomes:

$$\sum_{M_i \text{ sample point}} \frac{1}{\|r(\eta_i,\omega_i) - r(\mathcal{T})\|} \tag{15}$$

The time axis is also discretized: the deformation of the surface is implemented as an iterative process, the discrete time is simply the iteration number. With those discrete representations of (η, ω) and t, the derivatives involved in the final solution are approximated by the appropriate finite differences. In particular r_{tt} is given by a combination of the values of r at iterations t, $t-1$, and $t+1$: $r_{tt} = r(t+1) - 2r(t) + r(t-1)$. Replacing r_{tt} by its discrete approximation in the differential equation, we can express the surface at iteration $t+1$, that is the vectors $r(\eta_i, \omega_i)$, as a function of the surface at the two previous iterations t and $t-1$. If F is the right-hand side of 14, we have:

$$r(t+1) = r(t) + (r(t) - r(t-1)) + F \qquad (16)$$

After initialization, the deformable surface is iteratively updated using this relation.

To summarize, the algorithm can divided into two steps:

1. Initialize:
   - Extract the terrain features: shadows, discontinuities, normal discontinuities. Compute the polygonal approximations of the skeleton of the features.
   - Generate the discretization of the parameter space by computing a uniform sampling (η_i, ω_i) of the unit sphere.
   - Generate an initial surface. The initial surface is a sphere, that is $r_i = C + Ru_i$, where C is the starting point that is inside the object, R is the radius of the smallest object that we expect to extract, and $u_i = (r_i - C)/||r_i - C||$. The algorithm for selecting C is described in Section 5.
2. Iterate until the number of iteration is greater than t_0
   - For each point r of the surface, compute the projections $r(\mathcal{F})$ and $r(\mathcal{T})$.
   - Compute the derivatives of r with respect to η and ω using finite differences.
   - Compute the update term F using Equation 14.
   - Update the surface using Equation 16.

The result of the deformable surface algorithm is illustrated in Figure 11: The upper left part of the figure shows the features overlaid in white on top of the intensity image of a small scene. The upper right part shows a 3-D view of the terrain with the polygonal approximations of the features. The bottom three images show the evolution of the shape of the approximating surface as the algorithm proceeds.

5 Automatic object selection

We have assumed so far that a point is chosen inside each object to initiate the object segmentation process both in 2-D and 3-D. This point should be qualitatively "close to" the center of the object. The question of finding those initial points still remains. The simplest solution is to have an operator interactively select a point in the observed image. This would be acceptable in a teleoperated mode with the appropriate user interface. However, it would be more useful to be able to automatically compute the starting points from the input images. Since there is no prior constraint on where the objects may be in the scene, the only information that we can use are the features and a geometric model of the sensor. Specifically, the automatic segmentation is based on the observation that the presence of an object generates a shadow region in the range image. Therefore, the objects in the scene should be "near" the shadow regions extracted from the range image. The meaning of "near", that is the position of an object with respect to its shadow, is given by the sensor model.

The geometry of the problem is shown in Figure 12. For the sake of clarity, this geometry assumes a one-dimensional sensor; the reasoning can be extended without difficulties to a 2-D sensor: A projector P illuminates the scene while a camera C observes the illuminated scene. We assume that a sensor model provides the coordinates of P and C in a common coordinate system. An object in the scene creates a shadow region between points A and B, corresponding to illumination directions L_A and L_B that are known from the measured coordinates of A and B and from the sensor model. Based on this geometry, the occluding object must be within the dashed region R. A starting point for the 2- or 3-D snakes can be computed by taking the center of that region. It is important to note that this algorithm does not give us the center of the (unknown) object but rather a point that is enough inside the object for the object extraction algorithm to work.

The geometry is similar with a 2-D sensor except that the two points A and B are now contours. In practice, two corresponding points A and B are chosen on the shadow contour and the region R is identified using the 1-D geometry. The starting point S is selected within R at some nominal distance D from A. D is chosen based on the average expected radius of the objects in the scene and based on the minimum and maximum sizes of objects that we can handle given a gripper configuration. Those are reasonable criteria since there is no point in segmenting out objects that we cannot manipulate. D is also used to remove small shadow regions, presumably due to noise, and large regions, generated by objects too large to handle.

The key to automatic object extraction is an accurate geometric model of the sensor that allows us to compute the hypothesized position of objects in the scene based on observed shadow regions. We have implemented this technique using a model of our current sensor. However, it is important to note that the algorithm can be generalized to any range sensor provided that a geometric model exists. We are in the process of modifying the algorithm in order to use an existing geometric sensor modeling system [10]. This will lead to a largely sensor-independent segmentation program.

6 Manipulation

Once we have extracted object descriptions, either superquadrics or deformable surfaces, the last step is to grasp the object. Many different types of gripper design and grasping strategies are possible. The choice of a particular type of grasping is dictated by the analysis of the task. Assuming that the objects to be sampled are mostly isolated and are resting on a soft surface, *e.g.* sand, the grasping task has the following characteristics:

- The objects are far enough from each other. No collision occurs between the gripper and the neighboring rocks.
- We can allow the collision between the gripper and the neighboring sand. This is because
  - damaging the neighboring sand grains is not important,
  - the collision between the gripper and neighboring sand does not cause the configuration of the rock to change.
- we do not know the exact shape of a rock beforehand.

Based on the characteristics of the task and the possible grasping strategies [15], we have selected the spherical grasping strategy using a clam-shell gripper. The gripper has two hemispherical jaws that can close around the object. Using a surface representation of the objects, the grasping strategy is as follows: the center of the gripper is first aligned with the center of mass of the surface, then the gripper is rotated so that the jaws are parallel to the main axis of the surface. Finally the gripper is lowered until the jaws are in contact with the terrain surrounding the object. The object is grasped by closing the two jaws. Figure 13 show the gripper and the grasp operation.

This approach works well under the stated conditions. However, we need tighter control of the grasping operation than is provided by the spherical grasping in more difficult environments (*e.g.* Figure 4). In this case, we will use the object model calculated from the deformable surfaces algorithm conjunction with a three-finger gripper. The object model is more accurate than the superquadric model, and the three-finger gripper allows for more flexibility in the grasping. The price to pay is in longer computation time, and in more complex gripper design and control.

7 Conclusion

We have developed a testbed for sampling in unstructured terrain, that is the identification and manipulation of small natural objects. We have implemented the complete cycle of perception, representation, and manipulation. The objects are extracted from range images from surface features using either deformable contours or deformable surfaces. The objects can be represented by superquadric surfaces and by discrete surfaces. The system has been demonstrated in real natural environment using a manipulator equipped with a clam-shell gripper.

Our current work concentrates on building a more complete description of the terrain by using multiple images, hierarchical representation of the observed scenes, and by using more accurate object description such as deformable surfaces. We are working on a three-finger gripper to perform manipulation in a cluttered environment. Finally, we are exploring strategies for modifying the terrain using the manipulator to facilitate the sampling operations.

The sampling system currently resides on a small testbed. We want to eventually move it to a real vehicle, and to demonstrate the interaction between navigation and sampling, thus providing a complete system for planetary exploration.

References

[1] R. Bajcsy and F. Solina. Three Dimensional-object Representation Revisited. Technical Report MS-CIS-87-19, Univeristy of Pennsylvania, Department of Computer and Information Science, March 1987.

[2] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. An Autonomous Rover for Exploring Mars. *IEEE Computer*, (6), June 1989.

[3] P.J. Besl. Range Imaging Sensors. Technical Report GMR-6090, Computer Science Department, GM Research Laboratories, Warren, Michigan, March 1988.

[4] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.

[5] M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. In *Proc. 2nd International Symposium on Robotics Research*. MIT Press, Cambridge, MA, 1985.

[6] C. Brown. Fast display of well-tessellated surfaces. *Computer and Graphics*, 4(4):77–85, April 1979.

[7] J.D. Clinton. Advanced structural geometry studies, part I: polyhedral subdivision concepts for structural applications. Technical report, NASA, September 1971.

[8] C. Fox. *An introduction to the calculus of variations*. Dover Publications Inc., 1963.

[9] A. Gupta, L. Bogoni, and R. Bajcsy. Quantitative and Qualitative Measures for the Evaluation of the Superquadric Models. In *Proc. IEEE Workshop on Interpretation of 3-D Scenes*, November 1989.

[10] K. Ikeuchi and J. C. Robert. Modeling Sensor Detectability with VANTAGE Geometric/Sensor Modeler. In *Proc. of DARPA Image Understanding Workship*, pages 721–746. Science Application, Inc., May 1989. (a slightly longer version is available as CMU-CS-89-120).

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Intern. Journal of Computer Vision*, 2(1):321–331, 1988.

[12] Y.P. Lim. *Shape Recognition in the Rocks World*. PhD thesis, Dept. of Electrical Engineering and Computer Science, MIT, April 1988.

[13] A. Pentland. On the extraction of shape information from shading. Technical Report Vision science technical report 102, MIT Media Lab, March 1988.

[14] K. Sato, H. Yamamoto, and S. Inokuchi. Range imaging system utilizing nematic liquid crystal mask. In *International Conf. on Computer Vision*, pages 657–661, London, 1987.

[15] C.L. Taylor and R.J. Schwarz. The Anatomy and Mechanics of the Human Hand. In *Artificial Limbs*, pages 22–35, 1955.

[16] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Recognition. *Intern. Journal of Computer Vision*, 1(1):211–221, 1987.

[17] S.W. Zucker, C. David, A. Dobbins, and L. Iverson. The organization of curve detection: coarse tangent fields and fine spline coverings. In *Proc. Second Intl. Conf, on Computer Vision*, Tampa, December 1988.

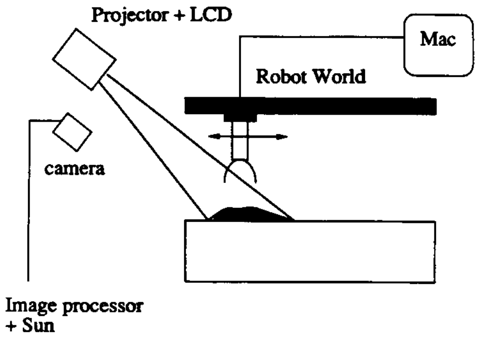


Figure 1: Testbed

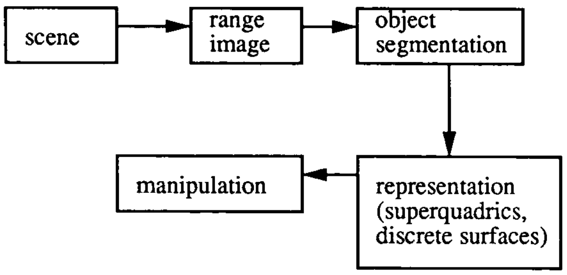


Figure 2: Control flow

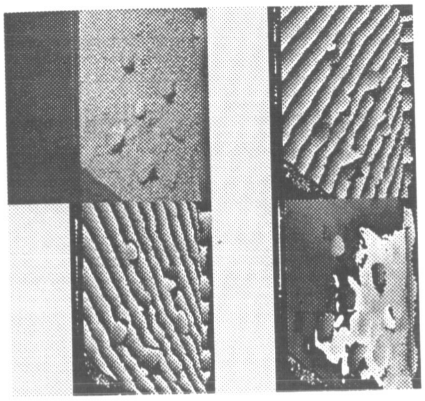


Figure 3: Range image of a scene

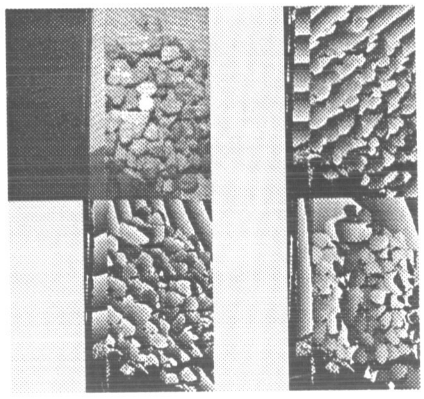


Figure 4: Range image of a complex scene

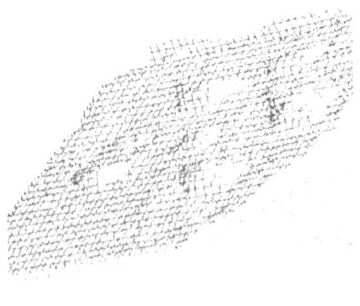


Figure 5: 3-D view of the data of Figure 3

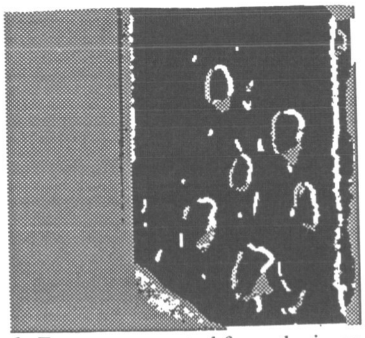
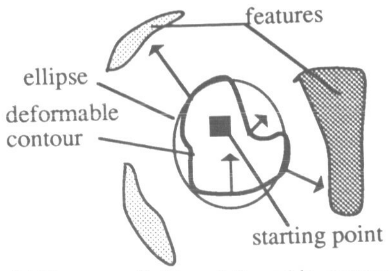
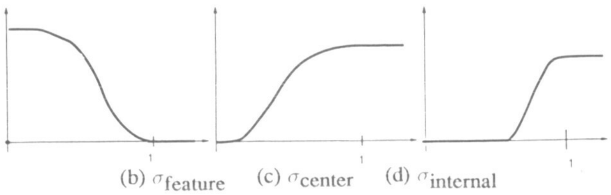


Figure 6: Features extracted from the image of Fig. 3



(a) Forces applied to a deformable contour



(b) $\sigma_{feature}$ (c) σ_{center} (d) $\sigma_{internal}$

Figure 7: 2-D segmentation algorithm

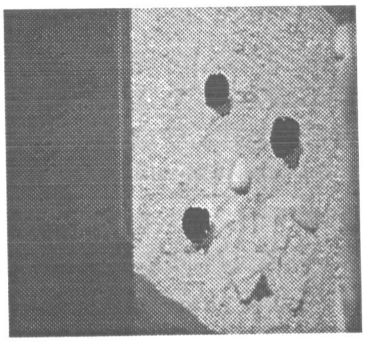


Figure 8: 2-D segmentation algorithm

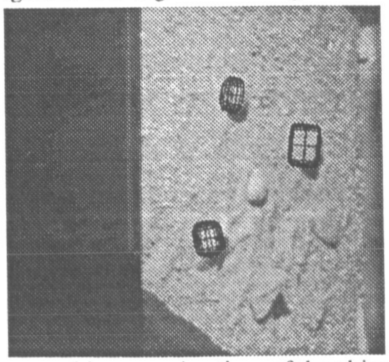


Figure 9: Superquadric approximations of the objects of Fig. 8

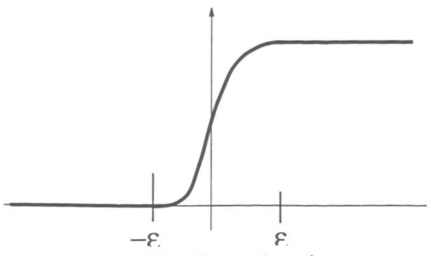
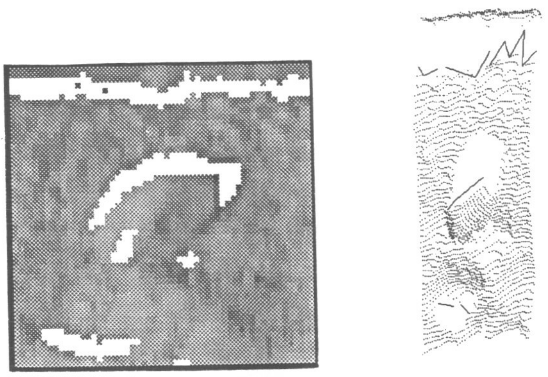
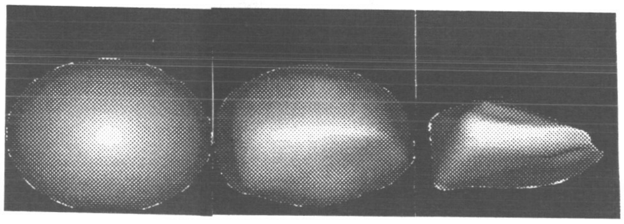


Figure 10: Sigma function



(a) Image of a scene and features (b) 3-D view of the terrain



(c) Initial shape (d) Intermediate shape (e) Final result

Figure 11: 3-D segmentation algorithm

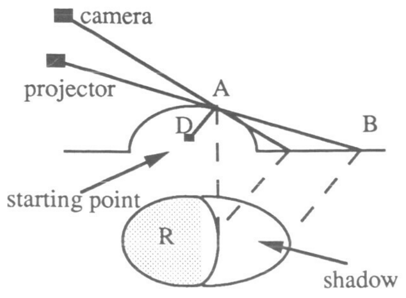


Figure 12: Automatic object selection

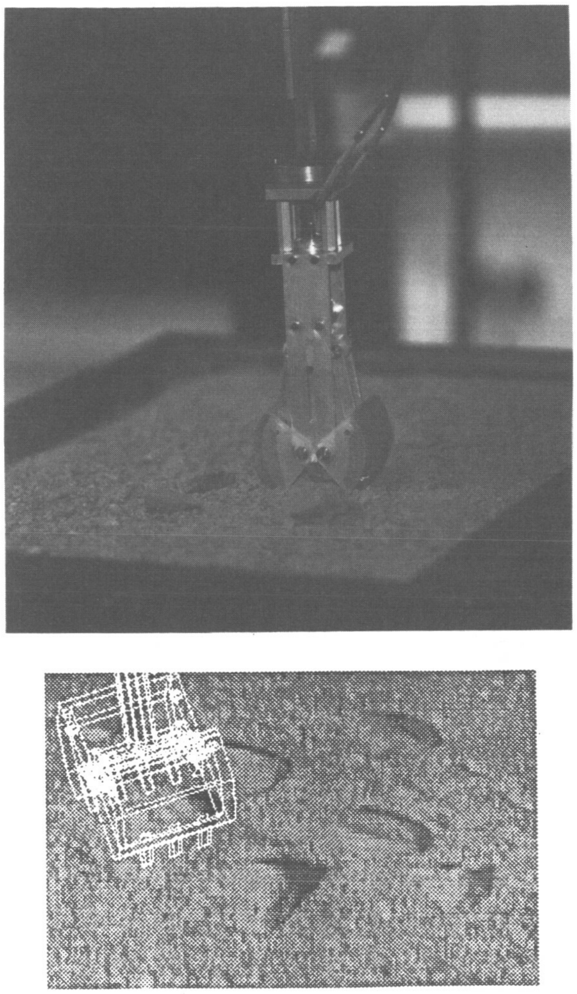


Figure 13: Gripper and grasp strategy

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

MISSION OPERATIONS

PRECEDING PAGE BLANK NOT FILMED

N91-20670

Advanced Automation in Space Shuttle Mission Control

Troy A. Heindel
National Aeronautics and Space Administration
Johnson Space Center

Arthur N. Rasmussen
MITRE Corporation

Robert Z. McFarland
UNISYS Corporation

Abstract

The Mission Control Center (MCC) at NASA's Johnson Space Center in Houston is certainly one of America's foremost technological achievements. From the early days of Apollo through Skylab to the Space Shuttle program, Mission Control has played an integral part in our ability to send humans into space and return them safely. Up until three years ago the technology of the MCC had remained virtually unchanged; flight controllers were supported by minimal tools and were expected through ponderous amounts of diligence and training to monitor the health of the country's leading aerospace products. The Real Time Data System (RTDS) Project was undertaken in 1987 to introduce new concepts and technologies for advanced automation into the MCC environment. The project's emphasis is on producing advanced near-operational prototype systems that are developed using a rapid, interactive method and are used by flight controllers during actual Shuttle missions. In most cases the prototype applications have been of such quality and utility that they have been converted to production status. A key ingredient has been an integrated team of software engineers and flight controllers working together to quickly evolve the demonstration systems.

Background

The Mission Control Center (MCC) has been the heart of NASA manned space flight operations since the Apollo program. It currently actively supports the Space Shuttle missions and will provide support for upcoming manned missions such as Space Station *Freedom* as well. The MCC is organized as a hierarchy of flight control officers headed by the flight director and organized into "disciplines" each of which monitors a specific portion of the Shuttle's onboard systems. The flight director is the leader of the flight control team and bears final responsibility for all mission decisions. Each discipline consists of a sub-team of controllers headed by a "front room" controller who supports the flight director and who in turn is supported by the "back room" controllers for the discipline. The organization of the MCC is shown in Figure 1.

In the past, the Mission Control Center (MCC) has relied exclusively on mainframe computers to process and display spacecraft data on monochrome display screens located in the flight control consoles. Although state of the art at the time of their installation, the systems have aged and now lag considerably behind current technologies. This is most evident in these systems' user interface which are clearly "user-unfriendly" by today's standards. The systems provide a primarily textual display of raw spacecraft data and require flight controllers to spend as much as 60% of their time converting raw data into the information needed to manage the mission[1]. Because of the low level of automation, a flight controller needs more than just a good understanding of the Shuttle's systems; the controller must spend many hours in simulated missions learning to quickly evaluate the raw data, build mental models that match the situation, evaluate them and come to a decision for the appropriate action. Developing the ability to perform these tasks in real time requires many hours of training and means a controller may spend as much as two or three years before becoming certified to support actual missions.

There are several additional factors that make the use of automated monitoring systems highly desirable in the MCC. NASA has a troublesome bi-modal age distribution in its personnel as shown in Figure 2. Due to a hiring freeze between the Apollo and Shuttle programs, there are two distinct populations of flight controllers consisting of highly-experienced Apollo-era veterans who are near retirement age and "Shuttle-only" flight controllers with less than five years of flight control experience. Although the Shuttle is possibly one of the most thoroughly documented pieces of hardware in the world, there is still a considerable body of uncaptured knowledge which only the Apollo veterans maintain. In each of the sixteen flight control disciplines, there are as few as one or two of these veterans remaining. The impending retirement of these veterans in the near future means the average experience level of most flight control disciplines will therefore diminish substantially.

Another contributing factor is the attrition level. Trained operations personnel are highly desired for new manned programs such as Space Station *Freedom*. Since the Shuttle program is the only source of such people, there is a natural migration of highly trained flight controllers to these new and exciting programs. The resulting high rate of attrition requires that new people be trained on a continuing basis, with the length of training time required further aggravating the situation.

The RTDS project was formed to meet the challenge of these problems. The guiding vision of the project is to demonstrate the use of advanced automation to improve the quality of real time flight decisions and thereby increase flight safety and mission success rates. Important components of this are the capture of knowledge, improvements in shortening training time and increasing its effectiveness, and containment of the growth of the size of flight control teams. The latter is especially important to providing operational support at affordable cost for long duration missions such as Space Station *Freedom* and manned planetary missions.

System Architecture

The applications within the RTDS project have been developed with three basic goals in mind: capture the knowledge and experience of expert flight controllers, decrease flight controller training time, and reduce the flight control team size. Much work has been done in laboratories on the design and implementation of advanced automation systems but in most cases the work remained unnoticed and isolated in the labs. Early on it was decided that the RTDS project would take the most mature of these technologies and demonstrate their use in the operational setting of the Mission Control Center. It was strongly felt that unless the technologies and techniques could be demonstrated in an actual operational setting, they would continue to encounter high resistance and slow acceptance due to the isolated and unproven nature of the laboratory systems.

This decision required that the RTDS system's architecture be designed for use in the operational setting. Because of the pressing demands of the active schedule of Shuttle flights, there has been a natural reluctance to modify existing operational systems to permit the testing of new technologies. This mandated that the RTDS systems would be independent of the existing mainframe-based flight control consoles and would operate in parallel with them. This parallel approach has yielded several unanticipated benefits. First is improved response time: the RTDS data acquisition system shaves 3 to 4 seconds from the 6 second data latency experienced by the existing mainframe system. Second, the existing system provides an immediately accessible standard against which the accuracy and effectiveness of the RTDS systems can be clearly and independently evaluated.

The need for an independent system produced a requirement for an end-to-end real time data system that could process the Shuttle's telemetry stream and deliver the data to demonstration applications for synthesis into information directly useful for flight control needs. The platform selected for the RTDS applications is a distributed environment comprised of Unix-compatible engineering workstations networked using the TCP/IP protocol. This environment was selected because of its flexibility, standardization and cost-effectiveness.

To support effective processing of real time data in this environment, a four layered architecture was designed. Each layer in the architecture plays a role in refining the data from a raw state into information. The layers are clearly defined and independent so that developmental evolution and testing can be performed in parallel. The architecture is shown in Figure 3.

In the first layer, data retrieved from a commercial telemetry processor travels by direct memory access (DMA) into a ring of raw data buffers maintained in a shared memory of the engineering workstation. Data is then removed from the ring, processed and finally placed into one of four application interface buffers, also resident in shared memory. Application programs in the workstation use library routines to retrieve the data from the interface buffers.

The raw data buffers are filled in rotation from the telemetry processor and are needed because the telemetry processor has extremely limited internal storage. The ring of buffers acts as a "rubber band" between the constant data rate coming from the telemetry processor and the subsequent processing of the data. This design is required to enable an operating system not designed for real time operations to support the continuous acquisition of data; the elasticity of the buffer ring compensates for the system load dependent rate of processor switching.

The telemetry processor performs the majority of decommutation prior to delivering the data to the workstation computer. The data is removed from the ring of buffers and processed to complete the decommutation of the data. The processed data is placed in an application buffer; each application buffer contains the data from one major frame of the telemetry stream and the buffers are used in round-robin rotation. (The Shuttle sends one major frame to the ground each second; each major frame contains a snapshot of the values of all onboard systems and sensors for that second.) Each time the application requests data, the application interface routines determine which is the most current application buffer and deliver data to the application from that buffer. The rotation of the application buffers allows an application to attach to a buffer and extract data from it without concern for the data being immediately overwritten.

The application buffers contain not only the data but also an indication of the "staleness" of the data. Each datum has an associated status that indicates whether that datum was received in the major frame contained in the buffer. This approach has been demonstrated to be much superior to the more common "current value table" (CVT) paradigm (in which the most recent value for each datum is made available without regard to the age of the sample). Although the CVT approach may be adequate for some situations, thorough analysis of shuttle telemetry requires time-homogeneity including the ability to determine how two values are related in time. Many situations cannot be properly analyzed with data values that are not bounded in time.

In addition to providing real-time telemetry data, layer one of the RTDS data acquisition provides a recording and playback facility that allows recording real time data as it is received. This has provided a major advance in capability for the flight controller: prior to RTDS, playback of real time data required the entire MCC facility be configured and operating. The RTDS playback allows flight controllers to review data independently at each workstation and has proven invaluable for several purposes. As an example, a recent launch was "scrubbed" just before liftoff due to a problem in the main engine area. Flight controllers were able to replay the data immediately after the scrub and doing so aided in quickly isolating the problem. This in turn allowed correcting the problem so the launch could be retried the next day, saving several days of extremely costly delay. The playback capability is also proving extremely useful for testing new applications as well as for verification and regression testing.

A tool has been developed to control the playback facility. It was dubbed "VCR" because its graphical interface has been made to closely resemble the remote control from a typical home video cassette recorder. The VCR tool allows playback of the data at varying rates, permits "rewinding" and "fast forwarding" and also allows replay points to be set so that the desired section of a recording can be repeatedly replayed automatically.

An Ethernet ™ distribution system has also been developed for RTDS that allows multiple workstations to receive real time data from a source workstation. The source workstation can be obtaining data from either a telemetry processor or from the playback facility. This permits multiple workstaions to share a single telemetry processor and provides redundancy since workstations receiving data from one telemetry processor can be reconfigured to receive data through a workstation instead.

The layer one software is written in the "C" language. It has been ported to several of the popular engineering workstations and additional porting is currently being performed.

The second layer of the architecture provides generic data manipulation which does not require domain-specific knowledge. This includes conversion of machine dependent floating point formats and calibration of raw data (PCM counts) into engineering units. This layer is also implemented using the "C" language.

The third layer supports domain-specific algorithms. This includes limit checking and calculations based on multiple parameter values. As part of the RTDS project a tool for building algorithm building tool called "CODE" (computation development environment). This tool allows non-programmers such as flight controllers to develop algorithms using a very high level, graphically-oriented language. CODE then translates the high level language into "C" code and links the algorithm to the real time data acquisition and workstation communication facilities within RTDS.

The fourth layer employs rule-based techniques to support both algorithmic and heuristic knowledge. Because of the real-time nature of RTDS, this layer is called upon only when third layer algorithms detect significant changes in the data values. A commercial off-the-shelf real time expert system shell, G2™ from Gensym Corporation, is used to implement the rules as well as an object-oriented graphical user interface.

All four layers communicate with each other and the flight controller through shared memory. The interfaces between the layers are designed to provide a high degree of visibility into the operation of the layers. This is important not only to facilitate testing but more importantly to provide the flight controller with the ability to examine the operations being performed. The latter is proving use for training and is a key ingredient in the acceptance of the RTDS system by experience flight controllers.

Development Philosophy

RTDS is an in-house project. Past experience has shown that direct user involvement is necessary in order to quickly deploy useful systems. For this purpose, the RTDS team is comprised of both development engineers and flight controllers. Several of the expert system applications have been developed primarily by flight controllers with occasional consultation with development support personnel. During the course of the project there has been migration of personnel between the areas resulting in a gain of strength in each.

Past NASA programs were forced in many cases to do ground-breaking engineering in areas such as processing of telemetry data. This approach is still necessary in some areas but can be avoided (at considerable savings in cost and development time) through the use of standardized, commercially available products. The RTDS project has demonstrated such use in several areas. Telemetry processing is done using a commercially available, fully-programmable telemetry processor. The computer hardware and operating system platform is Unix-based with plans for being based on the POSIX standards and new products such as operating systems that are Unix-compatible and provide true real time capability. Network communications are performed using TCP/IP and Ethernet; user interfaces operate under X-windows. The use of these standard products not only saves the cost and time of development, it also makes it possible to easily upgrade components to improve performance and take quick advantage of the cost effectiveness of new technologies.

Although the development strategy is suitable for producing useful applications quickly it does not guarantee that these same systems will be maintainable in the future. We chose to develop and or buy several tools which would ensure a high degree of maintainability. The G2 expert system shell has been extremely useful for this. The RTDS project has developed a set of standards which have been layered on top of G2 so that all applications built using the tool have the same look and feel. Additionally, the G2 tool has many knowledge management facilities which make maintenance an easier task.

Application Areas

The first application area selected was an expert system to support the Integrated Communications Officer (INCO). The INCO flight controllers monitor all communications systems on the Shuttle. As an initial area of investigation, the onboard payload communications system was selected as a system to be monitored by a rule-based expert system[2]. This system was used to monitor the payload communications system during the STS-26 mission, the first flight after the Challenger accident. The system represents several "firsts" in the MCC including the first use of a rule-based expert system and first use of a color graphics-based user interface in Mission Control.

One of the earliest and important RTDS applications created was an application that graphically monitors the Shuttle main engines and analyzes their performance. Just a few months prior to the launch of STS-26, analysis of data from test firings of Shuttle main engines showed flight controllers that certain conditions of main engine performance could lead to key engine valves "locking up". The data needed to diagnose the condition during actual missions was not available from the mainframe system and could not be made available for at least 6 months. As an interim, the controllers decided to read data from the console displays and manually enter the data into a personal computer which would perform the analysis to detect the condition.

The controllers also requested RTDS to examine the problem and propose a solution. Using the RTDS system, project personnel created a functional display containing nearly all the needed data in less than a week. By the time of the STS-26 launch, an application had been developed that performed the desired analysis and produced a graphical display as well. The application was certified for use in support of Shuttle missions and is currently in use during all Shuttle missions.

The RTDS project's Data Communications Officer Expert System (DATACOMM) is the first attempt in the MCC at position automation. Built using the G2 shell, DATACOMM performs all of the data monitoring tasks of the Data Communications Officer. The system currently does not yet send commands to the Shuttle but this is being considered. The data monitoring tasks include tracking data from Shuttle systems that is recorded on the onboard operational recorders as well as monitoring the health and status of related communications equipment. Once complete, DATACOMM will allow the merging of two flight control positions, reducing the INCO team from four persons to three. DATACOMM has been used during shuttle simulations with favorable results and will be tested during the STS-35 mission. To date, four person months have been spent developing DATACOMM. When finished it is estimated that a person-year will have been spent on development and testing.

The Jet-Control Expert System (Jet-Control) was developed for the Guidance, Navigation, and Control (GNC) officer. There are 38 primary Reaction Control System (RCS) jets on the shuttle which provide on-orbit attitude control. In the event that one or more of these jets should fail it is the job of the GNC officer to determine the control capabilities that have been lost. In the past, the GNC officer has used a time consuming twenty-five page paper procedure for making this determination. Jet-Control automatically makes the determination using telemetry data. Additionally, Jet-Control allows the GNC officer to perform "what-if" analyses with the remaining jets to quickly assess the remaining control capabilities and to do in-depth analysis of remaining equipment. During STS-31 (Hubble Space Telescope) Jet-Control detected the failure of three of the RCS jets; the GNC officer used the what-if capability to determine that the shuttle was one jet failure away from a loss of control in the +X direction (forward translation). Jet-Control was built in G2 in four months by one person.

The Remote Manipulator System (RMS), otherwise known as the Shuttle "arm", is vital to the success of missions such as the Hubble Space Telescope deployment. To aid the RMS flight controllers, RTDS personnel have developed a three-view display application for monitoring the position of the arm. Position monitoring is critical to ensuring that the arm is not over-stressed and that neither the arm nor any attached payload can collide with any part of the Shuttle. The application replaces a complicated off-line system that used a separate computer and three display screens and required a flight controller to manually enter each of the arm's multiple joint angles whenever they changed. The RTDS application has proven very useful and is very popular with the RMS controllers.

Visualizing the state of the Shuttle based on telemetry data is a problem faced by many members of the flight control team. Like the RMS arm position, the attitude and movement of the Shuttle is such a problem area. To demonstrate the potential of a graphical approach, an RTDS-based application has been developed that displays the Shuttle's flight instrumentation graphically. The display mimics the Shuttle's attitude and situation instruments and has been described by one astronaut as almost like being in the cockpit. The application is proving very useful for quickly and accurately determine Shuttle attitude and movement during all flight phases. It has also served as a demonstration of the proposed "glass cockpit" retrofit for onboard Shuttle instrumentation.

Of all flight control positions, the flight director is the most difficult. Filling the position requires a thorough knowledge of the Shuttle's systems as well as operational procedures. An RTDS application has been developed to assist the flight director with one of the more difficult tasks of the position, monitoring the weather at the launch site and the multiple possible landing sites around the world. Prior to the RTDS system, the flight director analyzed weather data chiefly by hand, with support from a weather officer. The RTDS application presents the sites on a display that shows the current weather in detail and indicates those data that area out of acceptable limits for ascent or landing. Reaction from flight directors has been very positive and are prompting requests for additional similar capabilities in other areas.

Technology Transfer

Much of the technology that has been developed by RTDS is being used by other data systems projects within NASA. The training division of JSC's Mission Operations Directorate is using the RTDS data playback capability to create standalone flight controller training. The per hour cost of a "full up" shuttle simulation is about $15,000. With an increasing flight rate it is more and more difficult to schedule enough training time to certify all trainee flight controllers. The stand-alone training capability will not only be cost effective, but will allow NASA to maintain an ample supply of certified flight controllers to meet the busy flight schedule.

NASA's Ames-Dryden Flight Test Facility, located at Edwards Air Force Base in California, employs the RTDS data acquisition system for telemetering the X-29 and F-18 research projects. The X-29 forward swept wing airplane requires timely (at least 100 times a second) monitoring and control of its control surfaces. The F-18 project is exploring the sparsely understood phenomena of "high alpha flight" or high angle of attack. The Air Force Test Flight Research center, also located at Edwards, is using RTDS data acquisition for the F-15 Short Takeoff and Landing (STOL) project in which modified jet engines are being evaluated as short takeoff and enhanced maneuverability options for the F-15.

The shuttle telemetry that is acquired by RTDS is distributed to other users besides flight controllers. RTDS has developed several data distribution methods which include direct memory access, Ethernet, and modem. Real time data can be displayed on office personal computers and is being used to evaluate the "office-based support" concept for the Space Station Control Center project. The data acquisition system of RTDS is being used by the Engineering Directorate of JSC to provide data for IMU testing and data archiving. The data acquisition system drivers and several of the user tools have been transferred to the Mission Control Center Upgrade (MCCU) project for incorporation into this larger upgrade effort.

Flight controllers of Mission Control have embraced the automation technologies which have been provided to them by RTDS. They have adopted these new tools into their flight controller tool boxes and have as a consequence developed new concepts for monitoring their systems. As past applications have been strongly based on established operations principals, future applications will continue to be. A new facility being incorporated into RTDS is a capability for applications to share information between workstations using network communication. None of the expert systems that have been developed in the MCC currently use this capability; they are stand-alone, isolated applications. This is extremely dissimilar to the actual functioning of the flight controllers who use them. The ability of flight controllers to function as a coordinated team is probably the most important single factor in the successful support of each mission. With the complexity of spacecraft increasing, the notion of team becomes even more important; no one person or machine can understand the system in its entirety. It is for this reason that in the coming months several of the stand-alone expert systems will be linked to each another. This linkage will undoubtedly spawn a whole new class of problems, but these problems must be overcome if we are to realize the full potential of this technology in Mission Control.

References

1. John F. Muratore, Troy A. Heindel, Terri B. Murphy, Arthur N. Rasmussen and Robert Z. McFarland, Space Shuttle Telemetry Monitoring by Expert Systems in Mission Control. In *Innovative Applications of Artificial Intelligence*, H. Schorr and A. Rappaport, Ed., AAAI Press, 1989, pp. 3-14.

2. Arthur N. Rasmussen, John F. Muratore and Troy A. Heindel, The INCO Expert System Project: CLIPS in Shuttle Mission Control. In *Proceedings of the First CLIPS Users Conference* to be published August, 1990.

3. D. A. Mackall, M. D. Pieckett, L. J. Schillin and C. A. Wagner, *The NASA Integrated Test Facility and Its Impact on Flight Research*, NASA Technical Memorandum 100418, Ames-Dryden Flight Research Facility, 1988, 14 pages.

Trademarks

Unix, G2 and Ethernet are trademarks of AT&T Corporation, Gensym Corporation and Xerox Corporation respectively.

Symbols and Abbreviations

COTS	Commercial Off the Shelf
GNC	Guidance, Navigation and Control
INCO	Integrated Communications Officer
JSC	Johnson Space Center
LAN	Local Area Network
MCC	Mission Control Center
MCCU	Mission Control Center Upgrade
MOD	Mission Operations Directorate
MMACS	Mechanical, Manipulator, and Crew Systems
RCS	Reaction Control System
RTDS	Real Time Data System

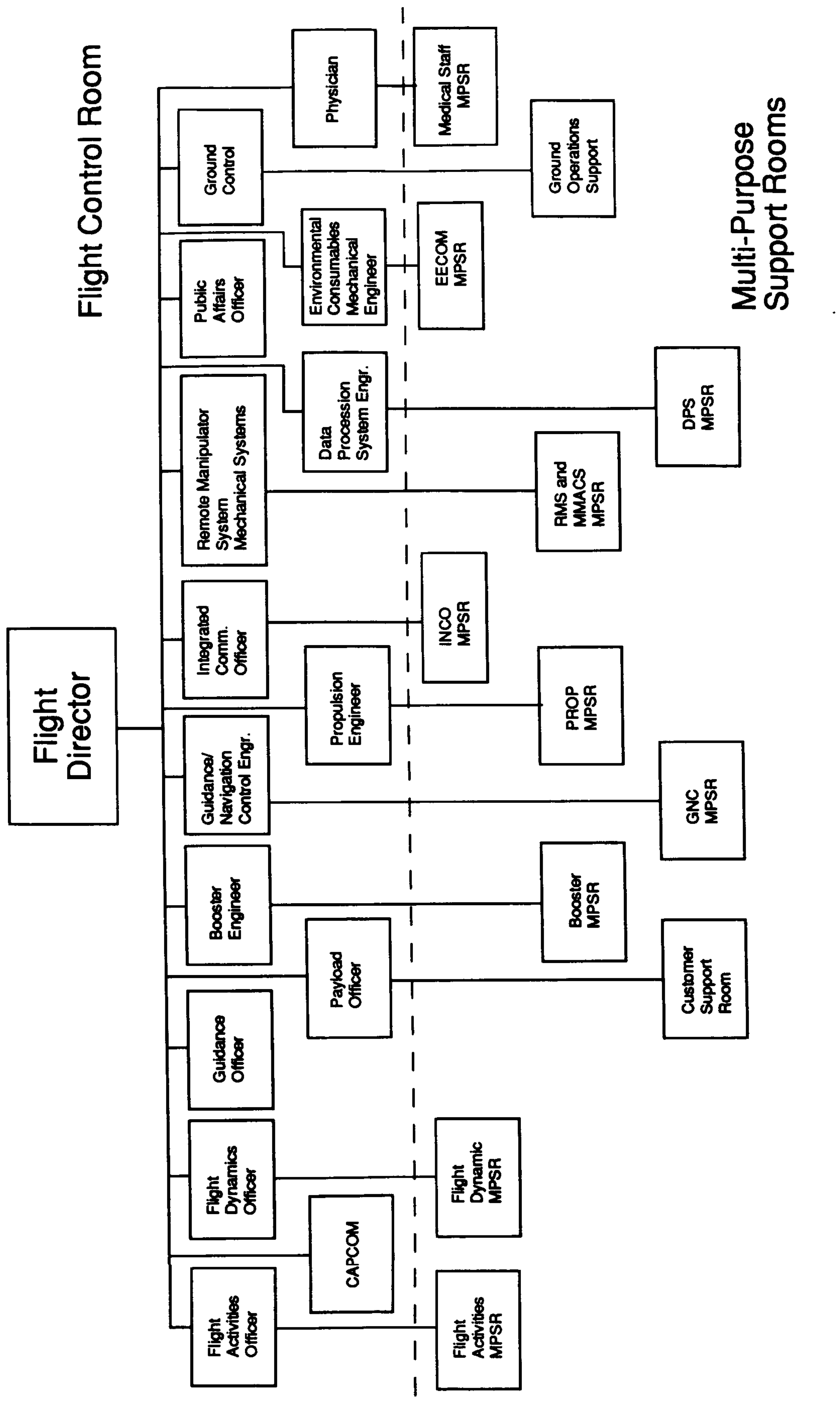


Figure 1. Mission Control Room Organization

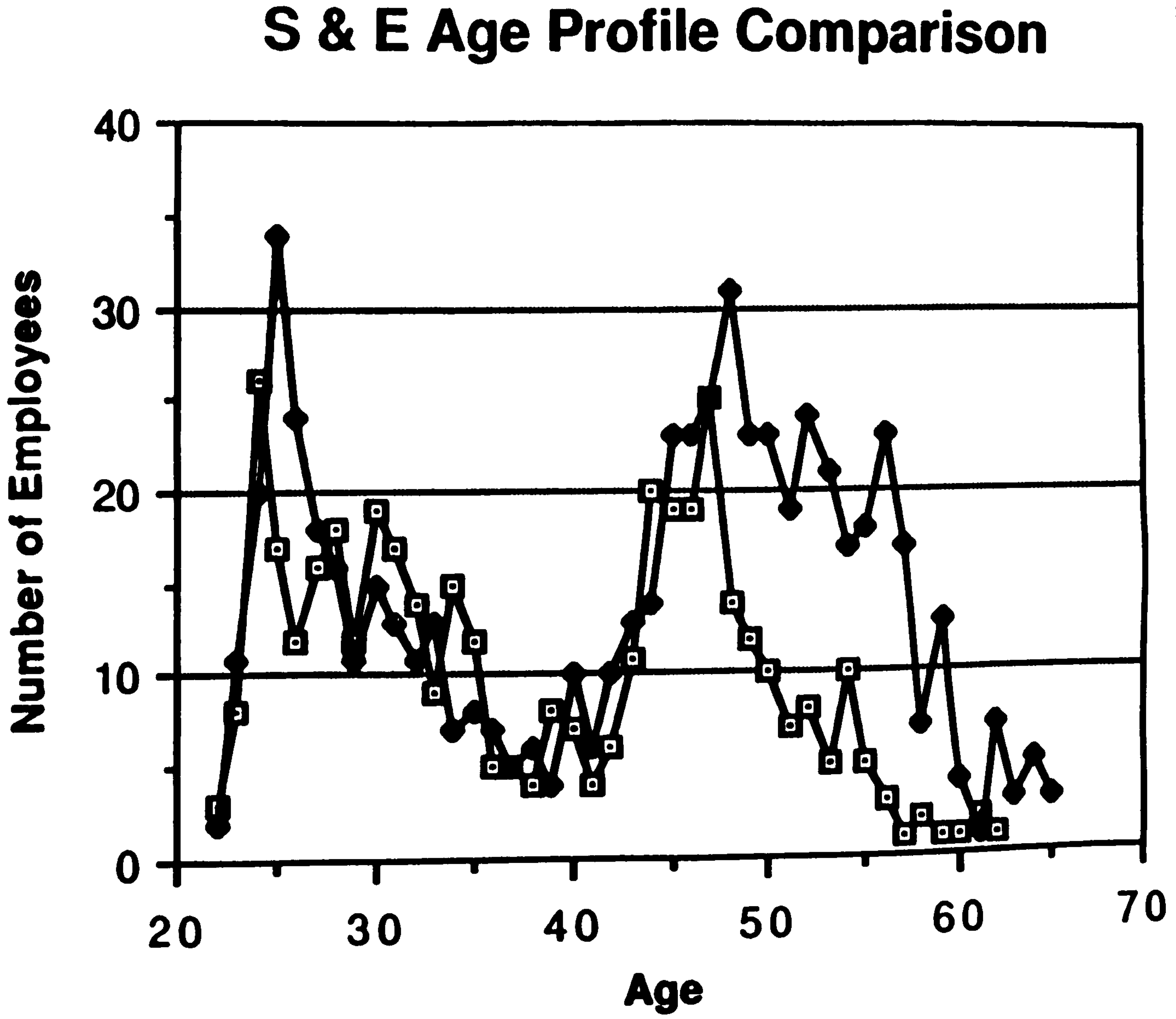


Figure 2. Bi-modal Age Distribution

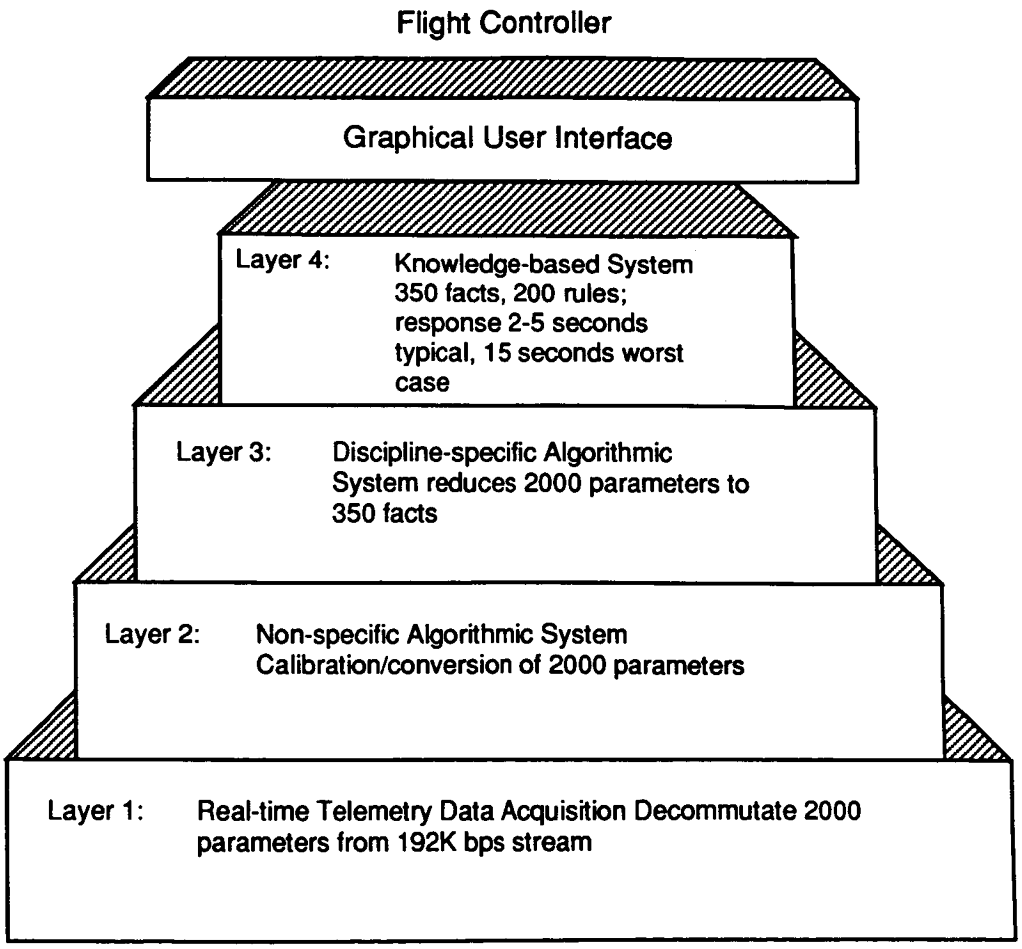

Flight Controller
Graphical User Interface
Layer 4: Knowledge-based System 350 facts, 200 rules; response 2-5 seconds typical, 15 seconds worst case
Layer 3: Discipline-specific Algorithmic System reduces 2000 parameters to 350 facts
Layer 2: Non-specific Algorithmic System Calibration/conversion of 2000 parameters
Layer 1: Real-time Telemetry Data Acquisition Decommutate 2000 parameters from 192K bps stream

Figure 3. RTDS Four Layer Architecture

N91-20671

SHARP: Automated Monitoring of Spacecraft Health and Status*

David J. Atkinson, Mark L. James, R. Gaius Martin

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

ABSTRACT

This paper briefly describes the spacecraft and ground systems monitoring process at the Jet Propulsion Laboratory and highlights some difficulties associated with the existing technology used in mission operations. A new automated system based on artificial intelligence technology is described which seeks to overcome many of these limitations. The system, called the Spacecraft Health Automated Reasoning Prototype (SHARP), is designed to automate health and status analysis for multi-mission spacecraft and ground data systems operations. The SHARP system has proved to be effective for detecting and analyzing potential spacecraft and ground systems problems by performing real-time analysis of spacecraft and ground data systems engineering telemetry. Telecommunications link analysis of the Voyager 2 spacecraft was the initial focus for evaluation of the system in a real-time operations setting during the Voyager spacecraft encounter with Neptune in August, 1989. The SHARP system will be delivered to the JPL Space Flight Operations Center for regular use by planetary flight projects, including the Galileo and Magellan spacecraft, and will also be applied to monitoring and control applications in the Deep Space Network's Network Operations Control Center.

2. INTRODUCTION

The Voyager 1 and Voyager 2 spacecraft were launched from Cape Canaveral, Florida, on August 20, 1977. The technology to monitor the health and status of these probes was designed and developed in the early 1970's. This now-antiquated technology, coupled with the heroic efforts of many JPL personnel over the last 13 years, has carried Voyager 2

* This paper was previously presented at SPIE, Orlando, Florida, April, 1990.

through near-fatal catastrophic events to four of our solar systems outer planets. Despite the spacecraft's failed radio receiver, sunlight damage to the photopolarimeter scientific instrument, and partially paralyzed scan platform (which houses Voyager's imaging system), JPL engineers have kept Voyager operational, enabling the capture and transmission of vast amounts of invaluable information and images of the Jovian, Saturnian, Uranian, and Neptunian systems.

During critical periods of the mission, up to 40 real-time operators are required to monitor the spacecraft's 10 subsystems on a 24-hour, 7-day-per-week schedule. This does not include the numerous subsystem and scientific instrument specialists who must constantly be available on call to handle emergencies. Unlike the 1980's, when JPL mission operations could focus on the two Voyager spacecraft, in the coming decade there will be an increasing number of planetary exploration spacecraft flying at the same time. In addition to the Voyagers, the Galileo and Magellan spacecraft have been launched in the past year and are now on their way to Jupiter and Venus, respectively. The Ulysses, CRAF (Comet Rendezvous and Asteroid Flyby), Mars Observer, and other spacecraft will follow in the next few years. To accommodate the increasing load on mission operations, JPL has established a Space Flight Operations Center (SFOC) to replace the individual mission control teams and spacecraft teams for each mission. A single, multi-mission flight team will operate all of the spacecraft. As more spacecraft are launched and begin to carry out their missions, the Space Flight Operations Center will require significant advances in automation technology in order to support the increasing workload on operations personnel and to ensure the safety of the spacecraft.

The Spacecraft Health Automated Reasoning Prototype (SHARP) was developed as part of an on-going effort to apply artificial intelligence (AI) techniques to mission operations automation. The primary task for an operational SHARP system will be multi-mission monitoring and diagnosis of spacecraft and ground systems in the Space Flight Operations Center. As tools such as SHARP are developed, they are demonstrated and evaluated in tough, operational settings to prove their performance. The Voyager 2 spacecraft was targeted for the initial demonstration of the SHARP system. The spacecraft's August 1989 encounter with the planet Neptune afforded an excellent opportunity to evaluate SHARP in a rigorous environment. The monitoring and troubleshooting of the telecommunications subsystem on-board Voyager 2 and the process of real-time telecommunications link analysis were selected as the initial operations functions to be automated. Telecommunications with the

Voyager 2 spacecraft suffers from frequent anomalies and requires coordination of monitoring and diagnosis efforts of both the spacecraft and ground telecommunications systems. Due to cumbersome and time-consuming manual processes and obsolete technology which will be discussed in later paragraphs, severe limitations exist on the current methods of analyzing Voyager telecommunications data. Even with the substantial improvement in computing support which is part of the new Space Flight Operations Center, the telecommunications area is both an operations area sorely in need of automation as well as one of the most challenging to automate.

3. TELECOMMUNICATIONS OPERATIONS

This section gives a brief overview of the telecommunications mission operations process, specifically focusing on the monitoring of spacecraft telecommunications subsystem health and telecommunications link status operations. Two of the major challenges for automation are described: The automation of manual data processing and data interpretation, and the automated real-time anomaly detection and analysis.

As noted earlier, each spacecraft is monitored on a continuous basis. To enable the receipt and collection of spacecraft engineering data, JPL operates three complexes of antennas located around the world. These complexes comprise NASA's Deep Space Network (DSN). With the exception of occultations and a short gap between two of the stations (Canberra and Madrid), a spacecraft is always in view from one of these Deep Space Stations (DSS), as the complexes are called. A scheduled observing period for a station is called a pass.

Three of the most important functions which are part of analysis of the telecommunications link between the spacecraft, Deep Space Network, and ground system computers at JPL are, 1) the numerical estimation of telecommunications subsystem and link performance, 2) the monitoring of real-time telecommunications activity and detection of failures or degraded performance, and 3) the diagnosis, isolation, and recovery from these problems. To accomplish each of these functions, a wide variety of information must be accessed and processed manually by an operator.

Predictions of telecommunications performance are embodied in a type of data known as "Predicts". Predicts are precise, numerical estimations of expected engineering data values for particular spacecraft and Deep Space Station parameters that impact the performance of the telecommunications link, such as signal-to-noise ratio and antenna elevation. Predicts are generated for each spacecraft pass over each ground station and can be divided into four categories: raw predictions, pass predictions, instantaneous predictions, and residual calculations. While the details of Predict generation and analysis are beyond the scope of this paper, it can be noted that much of the Predict calculation process is performed manually, and is tedious, time-consuming, incomplete, and error-prone. Telecommunications operators may spend up to two hours each day computing Predicts by hand using hardcopied listings of spacecraft activity, raw predictions, and pocket calculators. The SHARP system completely automates the process of Predict generation and analysis, saving up to two hours of operator time each day.

In addition to Predicts, telecommunications operators use the "Integrated Sequence of Events" (ISOE) to aid in monitoring telecommunications activity. The ISOE is a hardcopy listing of scheduled spacecraft and Deep Space Network activity. Operators use the ISOE in Predict calculations, alarm determination, and anomaly diagnosis. The operators must visually scan the ISOE to highlight relevant telecommunications information. This process is prone to error during periods of high spacecraft activity and when operators unknowingly do not reference the latest activity modifications to the ISOE. The SHARP system maintains a current, on-line database of ISOE information and automatically provides relevant telecommunication activity information from the ISOE to the system's other real-time monitoring processes and the operator as needed.

The monitoring of telecommunications and detection of anomalies is further complicated by the selection of alarm limits for spacecraft and Deep Space Station engineering parameters. Unlike the Predict values which are precise numerical predictions arising from a quantitative simulation of spacecraft performance, the engineering alarm limits are critical thresholds which define the acceptable range of engineering values on any telemetry channel. Excursions beyond the alarm limit range indicate imminent failure situations. In current Voyager spacecraft operations, alarm limits are determined manually according to the information in the ISOE, design information about spacecraft subsystem performance, and "rules of thumb" arising from the spacecraft team's experience with actual subsystem performance over the life of a mission. The current manual procedure to change alarm limits is so impeditive that for many engineering data channels typically a wide threshold is selected that incorporates the entire range of parameter conditions, thereby creating a risk of undetected anomalies. (See Doyle[1] for a discussion of problems in the determination of alarm limits).

In telecommunications as in other areas, the ultimate diagnosis, isolation, and recovery from failures, anomalous conditions, or degraded system performance often requires the intervention of experts who have years of specialized experience operating spacecraft subsystems (e.g., power, thermal, telecommunications). One of the most serious limitations on the current method of mission operations are the critical flight skills built up by these experts over the many years of flying spacecraft. These specialists must be on-call at any time, and are frequently consulted on a daily basis. The timeliness of an expert response to a problem can be critical in saving a spacecraft. Furthermore, when the experts retire, their critical skills are lost to mission operations. The Voyager 2 spacecraft has already been flying for almost 13 years, and is expected to operate until 2018. Many future spacecraft are expected to have similar longevity. The accumulated expertise of mission operations personnel is a critical resource which should be preserved, and not recreated every time a senior engineer leaves the flight project.

4. DESCRIPTION OF THE SHARP SYSTEM

The SHARP system applies artificial intelligence as well as conventional computer science techniques to automate and eliminate much of the tedious data processing and analysis associated with the monitoring of spacecraft and ground system health and status. Many of the manual, labor-intensive and error prone activities are eliminated in part or whole by SHARP. Some of these were described in the previous section. The major automated functions provided by the SHARP system include:

- Real-time anomaly detection and diagnosis;
- Visualization of channelized data and system status;
- Acquisition and centralization of engineering data in a single workstation;
- Real-time analysis of spacecraft performance predictions;
- Integration with specialized numerical analysis software, e.g., Fast Fourier Transforms for determining spacecraft antenna pointing accuracy.

Figure 1 illustrates a top-level view of the SHARP system. Shown are the individual modules that comprise the system, as well as relevant components that are external to the Voyager application of SHARP. SHARP is implemented in Common LISP on a Symbolics 3650 color LISP Machine. The system is currently being ported to a Sun workstation, also running Common LISP. SHARP relies extensively on an expert system building language called STAR*TOOL, developed at JPL[2]. The remainder of this

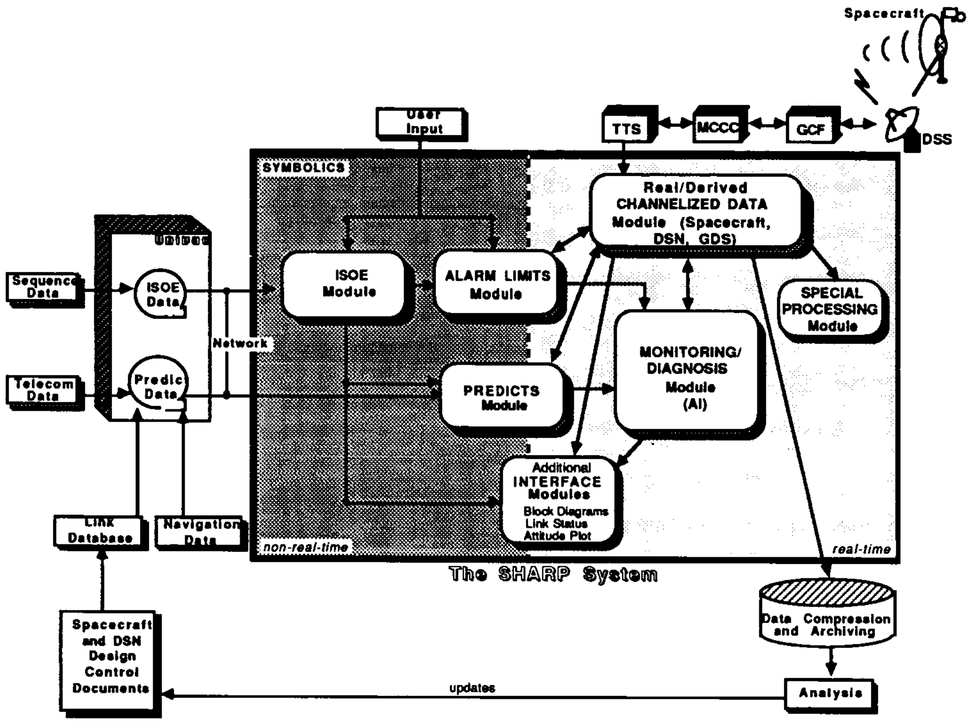


Figure 1. SHARP Telecom System Overview

paper will focus on the first of these automated functions: real-time anomaly detection and diagnosis. The remaining SHARP functions are described elsewhere[3].

In SHARP, the automation of fault detection and diagnosis is accomplished through the use of artificial intelligence programming techniques. Artificial intelligence techniques are distributed throughout all components of the SHARP system. Artificial intelligence programming methodologies have enabled more effective automation and thorough analysis for SHARP functions. Unlike the current manual methods used in space flight operations, fault detection and diagnosis in SHARP is extremely fast, taking approximately 1/200th of a second from receipt of anomalous data to determination of a diagnosis. This speed is directly attributable to the AI techniques incorporated by the design of the system. Some of the techniques used in the SHARP system include: Procedural reasoning, blackboards, reasoning using context trees, heuristic adaptive parsing, and spontaneous computation daemons. Figure 2 illustrates the

ORIGINAL PAGE IS OF POOR QUALITY

design of the "AI Module" in SHARP, which is responsible for fault detection and diagnosis.

4.1 Alarm determination

The first step in verifying nominal spacecraft performance is to determine whether received engineering data values are within acceptable limits. Data which is outside limits is considered in alarm, and must be explained. Data values can be classified as nominal, in "soft alarm" (possibly indicating a warning condition), and in "hard alarm" (possibly indicating an imminent failure condition). SHARP makes this determination automatically, by selecting the appropriate alarm limits for each channel of data and comparing new data against those limits in real-time.

The SHARP module responsible for this function is the Alarm Executive, as shown in Figure 2. The Alarm Executive module has a predetermined model of spacecraft states and transitions between those states. The Voyager application of SHARP has 39 such states. Alarm limits on each engineering data channel are determined in advance by the domain expert for each one of these spacecraft states. The limits are represented in table format, and organized hierarchically into a discrimination network 7 layers deep (the network is ultimately compiled into a a very efficient internal representation).

When a new engineering datum is received, the Alarm Executive first scans the Integrated Sequence of Events (ISOE) for the major activities and specialized activities which determine the spacecraft's current state, and further confirms the state by checking real-time engineering data related to spacecraft configuration. These are the keys used to search the spacecraft state discrimination network. In the case of the Voyager application of SHARP, the automatic gain control lock is checked to see if it is synchronized. The correct table of alarm limits is retrieved and the datum is matched against the appropriate alarm limits within the table after any additional conditions are checked, such as operator overrides. In general, more than a simple comparison of the datum against minimum and maximum threshold values is possible in determining an alarm condition. For example, an arbitrary function can be invoked to determine whether an alarm condition exists. These functions can break down the engineering datum into its individual bit status for example, or look at derivative information for trend detection.

In some cases, an anomalous spacecraft condition is directly indicated, e.g., based on error codes in the engineering data. In most cases, however,

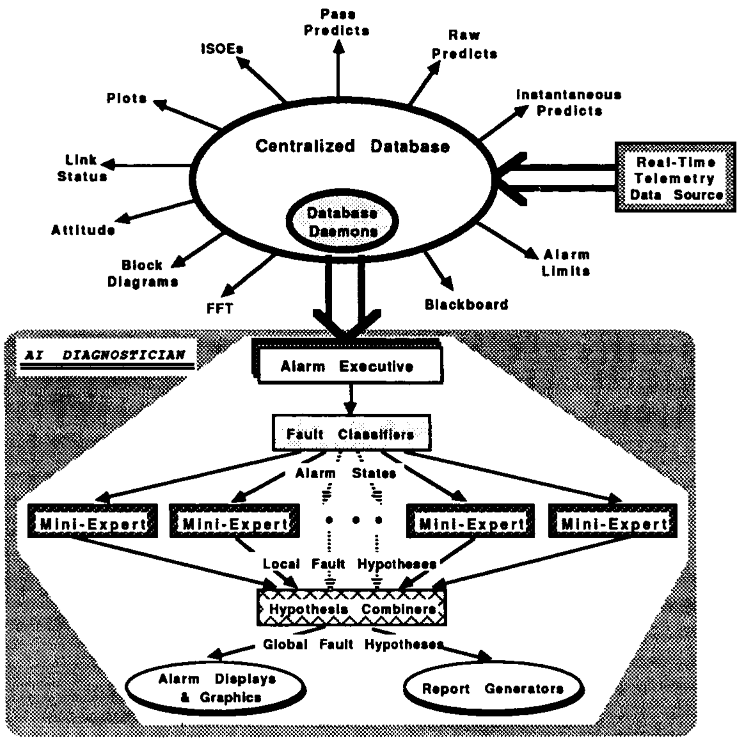


Figure 2. SHARP Artificial Intelligence Module

further analysis is required in order to determine the nature of the problem. The Alarm Executive makes this decision, and in addition monitors, logs, and reports to the telecommunications operator a number of attributes of the alarm situation, including the severity of alarm changes (i.e., from soft to hard alarm), the previous alarm status of the channel, and whether the operator has acknowledged the previous alarm messages. A variety of user interface and display changes are triggered by the Alarm Executive. If further analysis is required, the Alarm Executive informs the Fault Classifier module in SHARP. Analysis of alarm conditions by the Alarm Executive and Fault Classifier modules can proceed in parallel for any number of detected alarms.

4.2 Fault Classification

The Fault Classification module is a rule-based system which makes an initial interpretation of alarm conditions, spacecraft state, and the sources of engineering data indicating the anomaly. The result of the interpretation is a rough classification of the type of problem or its possible location in the telecommunications system, e.g., is it a spacecraft receiver problem, a possible configuration mismatch between the ground and spacecraft telecommunications subsystems, and so on. Frequently, there is no unambiguous interpretation available and subsequent diagnosis must proceed in parallel with several conflicting hypotheses. The products of the fault classification are asserted into a database which results in a pattern-directed invocation of specialized diagnostic routines, called "Mini-experts", described below. This architecture of hierarchical invocation of specialized diagnostic knowledge is related to the paradigm of cooperating specialists in classificatory diagnosis embodied in the CSRL system[4] and in the StarPlan system[5].

4.3 Mini-expert diagnostic routines

The Voyager telecommunications application of SHARP includes approximately 40 mini-experts. These specialized diagnostic routines are each responsible for the local diagnosis of a specific fault or class of faults, such as particular channels in alarm, conical scan errors, configuration mismatches, or loss of telemetry. Mini-experts can be either cooperating or non-cooperating. A non-cooperating mini-expert focuses only on its designated fault area, and generally its conclusions can and should be reported independently to the operator. A cooperating mini-expert has the additional capability of searching beyond its local area to identify related faults that are likely to occur. In the process of this search, the cooperating mini-expert triggers other mini-experts who are specialists in those related areas. Information is exchanged between the mini-experts using a blackboard message system.

Mini-experts encode a procedural network of diagnostic decisions and analyses. They are related to rules in the Procedural Reasoning System (PRS) of Georgeff and Lansky[6], although the representation mini-expert procedures differs. Mini-expert rule definitions include high-level descriptions of preconditions, activation and execution contexts, spacecraft state descriptions, relevant real-time data sources, hypotheses, and sequences of analyses and decisions which are part of the diagnostic process. Mini-expert knowledge definitions are not interpreted by SHARP. Instead, SHARP contains a compiler which generates Common LISP code from mini-expert descriptions and automatically installs the definitions into the SHARP run-time environment. The compiler performs the necessary bookkeeping and also checks for consistency with the other, installed mini-experts. Currently, a trained knowledge engineer must develop mini-expert definitions by hand, and this constitutes a bottleneck for application of the system. To aid in knowledge acquisition, we are developing a graphical interface, called a "visual rule-building system" which can be used by domain experts to create mini-experts which would then be directly compiled by SHARP as before.

As mentioned above, the Fault Classification module may not determine a unique mini-expert to invoke. In this case, multiple mini-experts are invoked which pursue diagnoses in pseudo-parallel. Pseudo-parallelism is implement in SHARP using facilities provided by STAR*TOOL, which includes parallelism as a fundamental control structure. The various mini-experts and their rules operate in isolation of one another by executing in independent contexts[7] provided in the STAR*TOOL memory model. Contexts can be organized into a tree-like structure to represent contradictory information resulting from changes in facts or from the introduction of new or contradictory hypotheses.

4.4 Hypothesis Combination

The Hypothesis Combiner module has the role of combining multiple fault hypotheses generated when several mini-experts are invoked in parallel by the Fault Classification module. The module communicates with mini-experts through SHARP's blackboard. Related fault hypotheses are combined into a single, more encompassing explanation for the operator (e.g., when there is a single action to take in response). Redundant hypotheses are eliminated in the process as well. When there are conflicting explanations for a detected problem, SHARP presents all of the explanations to the operator along with the separate recovery recommendations. In some cases, the operator is privy to information and knowledge which SHARP does not have, and can effectively disambiguate the situation. In any event, the final problem determination step and any corrective actions is left to the operator in cases of ambiguity.

5. VOYAGER ENCOUNTER WITH NEPTUNE EVALUATION

Approximately one month before the Voyager encounter with the planet Neptune, a Symbolics workstation with SHARP loaded on it was moved from the Artificial Intelligence Laboratory at JPL into the real-time telecommunications operations area for the Voyager spacecraft. There were severe restrictions on how SHARP could interact with other Voyager systems. To simplify the installation, SHARP obtained spacecraft engineering data from the Voyager Test and Telemetry System over a printer port. Unabridged Integrated Sequence of Events and raw Predict data were loaded into SHARP using tapes, rather than through network connections as in the Artificial Intelligence Laboratory.

During the demonstration period, SHARP helped find the cause of a Voyager science data error anomaly which appeared in the telemetry from the spacecraft as an excess error count. The SHARP system's graphical displays were used by telecommunications personnel to identify the problem and to characterize its magnitude. The problem was isolated using SHARP and other, manual trouble-shooting techniques to the Voyager ground data system and was corrected by the replacement of a wide-band interface unit in the Voyager Data Acquisition and Capture System (DACS). SHARP helped verify that the replacement of the unit actually fixed the problem. In a matter of hours, SHARP was able to assist operators in solving an anomalous condition which could have easily escalated to a more serious problem during the encounter itself, and could have taken human operators days or weeks to isolate without SHARP.

Also during the demonstration period, the knowledge engineer of SHARP and the domain expert would review alarms that SHARP had given. Generally, these alarms were correct. In one alarm situation, SHARP was giving warnings about the loss of the telecommunications signal. This ultimately turned out to be a false alarm as the spacecraft was undertaking a particular maneuver that the SHARP knowledge base did not contain, thereby leading the diagnostic system into an erroneous conclusion about antenna pointing. In other cases, SHARP was able to detect conditions where the Deep Space Station antenna tracking the spacecraft was drifting off point. SHARP detected these problems in a matter of seconds, and reported the condition to the telecommunications operators. Unfortunately, due to their previous lack of ability to detect and diagnose antenna pointing problems, the real-time telecommunications operators at JPL did not have procedures for alerting the Deep Space Station operators (possibly on the other side of the world) to antenna drift situations detected by SHARP. When the antenna drift reached a sufficient magnitude and urgency for the station operators to notice and correct, SHARP was able to detect the resolution of the problem and cancel the alarm situation. SHARP detected and correctly diagnosed other non-critical problems with the receiver automatic gain control and the S-band travelling wave tube temperature on board the spacecraft.

On the whole, the encounter with Neptune went extremely smoothly for the Voyager spacecraft. SHARP did not get a chance to make any really dramatic diagnoses, and the diagnostic system described in this paper did not get a strenuous operational test. This underscores the difficulty in testing the diagnostic ability of real-time monitoring and control expert systems in operation settings: you may not get any problems! Using simulated data (based on historical problems with the spacecraft and based on synthetic situations) we were able to test SHARP much more thoroughly in the laboratory. SHARP is able to analyze 39 classes of telecommunications problems, and make about 60 unique diagnoses which require some problem-solving by the mini-experts to determine. Another 20 telecommunications problems are detectable by SHARP, but can be reported directly to the operator. Our domain expert estimates that SHARP covers approximately 80% of the known types of faults experienced in spacecraft telecommunications for Voyager. The remaining 20% include diagnoses which could be made if SHARP had the appropriate real-time data and additional knowledge engineering. As with most complex systems, there is always the possibility of novel faults. SHARP does not have the ability to successfully diagnose and explain a novel type of fault (nor was it intended to), but we are confident in the system's ability to detect departures from expected, nominal behavior.

6. EXPECTED BENEFITS FROM SHARP APPLICATIONS

There are four principle areas where the JPL telecommunications users of SHARP expect to see benefits from application of the system and its descendents, which we are now developing. These areas are safety, workforce savings, reliability, and productivity.

Through its accurate detection, analysis, and tracking of the antenna drift and pointing conditions during the encounter, SHARP showed that it can detect and analyze important problems in a matter of seconds which currently take human operators minutes or hours. This provides an extra margin for ensuring the safety of the spacecraft, and thereby supports the success of the mission as a whole. The SHARP Voyager telecommunications domain expert, a man with over 20 years of experience who has cognizance not only for Voyager telecommunications operations but for other spacecraft as well, as stated publicly that the Soviets would not have lost the first Phobos spacecraft if they had SHARP applied to their telecommunications. One of the stated causes of the loss of the Phobos spacecraft has been that the spacecraft antenna drifted until the telecommunications link was lost due to a faulty attitude control command.

A second major benefit from application of SHARP will be in the area of workforce savings. Through its automation of many manual functions, SHARP promises to reduce the real-time link analysis operations staff by a factor of five, and there is reason to believe that similar savings may be possible in other operations areas. This is precisely the type of benefit from automation which is necessary to support the single multi-mission flight team in the new JPL Space Flight Operations Center.

The system-wide status monitoring afforded by SHARP, and not discussed in detail in this paper, helps operators assure correct telecommunications system configuration. This is expected to reduced the number of commanding errors to the spacecraft and ground systems, and thereby reduce the loss or corruption of data due to configuration problems.

Finally, the SHARP system is expected to enhance the productivity of operations personnel by freeing them from the tedium of watching raw data and interpreting it for themselves. SHARP shifts the burden of routine monitoring operations, and most of the boring, manual computations which are involved, away from the operator to itself. This will enable operations personnel to perform required analyses more efficiently, and to exert a higher level of "supervisory monitoring" over multiple spacecraft subsystems on multiple spacecraft.

7. CONCLUSIONS

Spacecraft and ground data systems operations present a rigorous environment in the area of monitoring and anomaly detection and diagnosis. With a number of planetary missions scheduled for the near future, the effort to staff and support these operations will present significant challenges.

The SHARP system was developed to address the challenges of automation in a multi-mission operations environment by augmenting conventional automation technologies with artificial intelligence. Its successful development and demonstration have led to a number of important conclusions. First and foremost, artificial intelligence technology is ready for application to spaceflight operations. The techniques can be used alongside conventional computer science techniques, and diagnostic knowledge-based systems can be embedded in the resulting application system. Acceptable real-time performance can be achieved. SHARP was never pushed to the limit of its speed or memory resources; in fact, most of its time was spent idle, waiting for new engineering data to process. This gives us confidence for broadening the approach in SHARP to multiple spacecraft subsystems.

The evaluation by Voyager personnel also taught us that the types of automation provided by SHARP are high desired by operations personnel, and are not viewed as job-threatening (although they may be in some cases). Operators were able to readily use the system with minimal training, and were enthusiastic about using the wide variety of graphical displays and options.

SHARP is now being extended and developed to a higher level of readiness so that flight projects such as Voyager, Magellan, Galileo, and others can use it directly. The system will be completed in 1990 and delivered to the Space Flight Operations Center for further evaluation and application to Magellan telecommunications. Separately, SHARP is also being applied to the Deep Space Network, Network Operations Control Center at JPL, with an operational system planned for 1991. Applications for remote monitoring and control of spaceborne instruments and experiments are also under consideration.

8. ACKNOWLEDGEMENTS

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors wish to acknowledge Denise Lawson and Harry Porta for their contributions to SHARP development, and to Boyd Madsen for providing expert knowledge for Voyager telecommunications. Portions of this work have been reported previously while in progress at the 1989 Goddard Conference on Space Applications of Artificial Intelligence, Greenbelt, Maryland, May 16-17.

9. REFERENCES

1. Doyle, R., Sellers, S., and Atkinson, D., "Enhancing Aerospace System Autonomy Through Predictive Monitoring", *Proceedings of AIAA Meeting on Aerospace Sciences,* American Institute for Aeronautics and Astronautics, Reno, Nevada, 1989.

2. James, Mark, and Atkinson, David, "STAR*TOOL -- An Environment and Language for Expert System Implementation", *Jet Propulsion Laboratory Report NTR C-17536,* August, 1988.

3. Lawson, D., and James, M., "SHARP: A Multi-Mission Artificial Intelligence System for Spacecraft Telemetry Monitoring and Diagnosis", *1989 Goddard Conference on Space Applications of Artificial Intelligence,* pp. 185-200, NASA, Greenbelt, Maryland, 1989. Also *Jet Propulsion Laboratory Publication 89-23,* May, 1989.

4. Chandrasekaran, B., "Distributed Knowledge-based Systems for Diagnosis and Information Retrieval", *Department of Computer and Information Sciences Report 763180/714659,* November, 1983.

5. Siemens, R.W., Golden, M., and Ferguson, J., "StarPlan II: Evolution of an Expert System", *Proceedings of the Fifth National Conference on Artificial Intelligence,* American Association for Artificial Intelligence, Philadelphia, PA, 1986.

6. Georgeff, M., and Lansky, A., "A System for Reasoning in Dynamic Domains: Fault Diagnosis on the Space Shuttle", *Technical Note 375,* Artificial Intelligence Center, SRI International, January, 1986.

7. McDermott, D. V., "Very Large PLANNER-type data bases", *Memo AIM-339,* AI Laboratory, Massachusetts Institute of Technology, 1975.

N91-20672

Challenges in Building Intelligent Systems for Space Mission Operations

Wayne Hartman
Artificial Intelligence Group
Ford Aerospace Corporation
Sunnyvale, California

ABSTRACT

The purpose of this paper is to provide the reader with a top-level look at the stewardship functions performed in space operations, and to identify the major issues and challenges that must be addressed to build intelligent systems that can realistically support operators in performing complex space operations functions. The focus is on decision support activities involving monitoring, state assessment, goal generation, plan generation, and plan execution. The bottom line is that problem solving in the space operations domain is a very complex process. A variety of knowledge constructs, representations, and reasoning processes are necessary to support effective human problem solving. Emulating these kinds of capabilities in intelligent systems offer major technical challenges that the artificial intelligence community is only beginning to address.

INTRODUCTION

The world of military space mission operations is rapidly transitioning from a research and development focus to a truly operational focus ready to support a variety of peacetime and wartime objectives. As contractor engineers and experienced operators are replaced with less experienced "blue-suit" operations personnel, intelligent decision support capabilities must be developed to offset the loss of expertise and experience. The remainder of this paper provides an overview of the functions performed in space operations; discusses the difficulties and challenges of providing robust problem-solving support to space operators; and presents top-level architecture components for addressing some key problem-solving activities.

TOP-DOWN LOOK AT SPACE OPERATIONS

The space operations job involves remote monitoring and control of a complex space system to accomplish a variety of mission objectives. Operators must maintain the space system in the best state, configuration, and health possible to support maximal mission accomplishment both in periods of high demand and over the entire life of the space system.

Because operators are physically removed from the space systems they monitor and control, they must constantly create and deal with a perceived system state derived from incomplete snapshots of telemetry. Uncertainty, primarily a result of discontinuous monitoring of limited telemetry, and the need to make critical decisions under time- and information-restricted conditions, greatly magnify the complexity of space operations decision making and problem solving.

The basic operations functions, whether dealing with an entire space system or a specific subsystem, are to:

Monitor -	Observe indicators/telemetry from the system. Perform analysis to derive other attributes or state information as needed.
Assess -	Determine the system state and decide if action is required to improve that state.
Plan -	**Fault isolate and perform causal analysis to focus problem-solving activity. Construct goals and find actions that support those goals.**
Act -	**Decide on specific action and perform that action. Monitor effects and reassess, replan, or take other action as necessary to meet objectives.**

Monitoring is a dynamic, discontinuous process. The analyst only has access to snapshots of telemetry and, due to time and frequency constraints, must focus on what telemetry parameters to look at in a given situation. This focus dynamically changes as the analyst uncovers indications that something is possibly anomalous. To further complicate this process, telemetry is generally incomplete, noisy, and subject to occasional dropouts. This results in a great deal of information uncertainty. Behavioral and environmental uncertainty introduce added difficulty to understanding what is going on. Some aspects of system state can be derived by analyzing the relative behavior of groups of telemetry parameters over time. The key objectives of monitoring are to verify that intended actions are accomplished correctly, and to observe system health and status so that problems can be recognized and addressed expediently.

Assessment relies on monitoring to provide an observed perspective of current system state. Depending on the situation at hand, different viewpoints are used to define the current state. These viewpoints help to focus analysis resources on the areas most important to maintain correct system behavior. In the Space Mission Support (SMS) environment, space systems have mission objectives and other derived system objectives that allow optimum mission performance over the design life of the space system. Assessment involves comparing the capabilities of the system in its current state to the capabilities the system is desired to provide for optimal, overall mission performance.

Planning relies on assessment to provide a focus on what desired capabilities are sub-optimal/ unsatisfactory in the current system state. Using detailed system knowledge and current state information, fault isolation and causal analysis are applied to identify suspect problem states and behavior. System knowledge, at many levels, is then used to establish goal states that better provide desired system capabilities. Once goal states are generated, planning can search for events or actions that cause the system to transition from the current state toward the desired state. Planning must also determine the ramifications or expected side effects of specific actions to the degree possible. Detailed models are required to support this process, accompanied by judicious use of simulation.

Acting relies on planning to provide options for action along with their expected results including any consequences and side-effects. An optimum course of action is decided upon, executed, and monitored to assure results are in line with expectations. If not, alternative actions are selected or the process backs up to the monitoring, assessing, or planning phases. Deciding on an optimum course of action is not an easy process. There are time constraints that limit how much analysis can be done. In addition, it is not easy to weigh the impacts and benefits of different options against each other. The importance of key factors varies significantly with system state, current mission objectives, and overall space system health.

Uncertainty muddies the entire process. Perceived system state is never complete or exact. It is a best guess based on what we can observe. System models are accurate only to the level they are modeled. Finding and quantifying side effects is not easy. Simulation can be used to help, but complete simulation is too costly in a time-constrained environment. Effects monitoring must be focused to yield timely and useful information, but this focusing may prevent critical impacts from being found.

Difficulties and Challenges

From the above discussion, many difficulties and challenges obviously confront anyone attempting to build intelligent systems to emulate human problem-solving and decision processes in the space mission support environment. The primary challenges involve different kinds of knowledge that must be represented, providing reasoning operators that can act on this knowledge in a variety of ways, and developing dynamic, flexible control structures and mechanisms for controlling these reasoning operators in a manner that results in useful and effective reasoning, decision, and problem-solving processes. Some key areas that present tough challenges include:

Compositional Behavior and Focus: For many systems and subsystems, the specific functions performed by the parts are dependent on some overall system state. This system state may be influenced by the outside world and/or by the combined states of its parts. In any composition, there may be behavior that can only be represented at these higher composition levels. In the Space Mission Support (SMS) domain, important behavior occurs at many levels. An analyst may have to understand the behavior of all levels, and will move his focus up and down as necessary to accomplish his objectives in an effective manner. In many cases this requires integration of information available from several levels.

Multiple Viewpoints and Cooperating Agents: An analyst looks at a system from a given point of view. This viewpoint emphasizes certain characteristics or behavior for the purpose of making it easier to categorize system state and reason about problem solving from the analyst's perspective. In the SMS domain, the primary mode of operation is to have several specialists monitoring system behavior from their unique perspectives, cooperating and interacting as necessary to identify and resolve any problems that arise. To further complicate matters, each specialist may have several viewpoints he selects from depending on what is happening and where he is in the problem-solving process. Any intelligent system addressing the SMS environment must include a representation structure for these viewpoints, allowing the reasoning processes to select, focus on, and change particular viewpoints as appropriate for the problem-solving process.

Depth of Model and Abstractions: For any model, choices must be made regarding the depth or level of detail of the model in the various areas in which the model applies. In the case of a space system, each subsystem is modeled to the depth necessary to meet the objectives for the overall model. These objectives should specifically support the uses for which the model is employed. For many specific objectives, acceptable decisions can be made without resorting to a deep model. Higher level abstractions of behavior and states can provide sufficient detail at a much lower computational cost.

Also, the simplification provided by the abstraction can make it much easier to control and guide the reasoning process. For efficient decision making in the SMS environment, these abstraction layers must be supported in the knowledge representation and also be dynamically accessible to reasoning processes.

Problem-Solving Environment Components

The remaining sections outline some key knowledge representation, reasoning function, and reasoning control issues for five critical components of any problem-solving environment: State Determination, Situation Assessment, Problems Construction, Goals Construction, and Plans Construction.

State Determination: State determination involves building a perceived current state of the system/world. The current state consists of the states of all components or concepts, and current values for all attributes or parameters. Some values and state information are directly reported in telemetry from the system. Many others can be derived by observing the reported values over time and matching this behavior with knowledge from the system and world knowledge bases. The key elements required to support this process are shown in Figure 1. They include:

a. **TLM:** Telemetry from the system/world that provides direct information about current state.

b. **World K, System K:** Detailed knowledge about the system function, design, and behavior with respect to the world environment.

c. **Previous System State:** Last known state of the system/world.

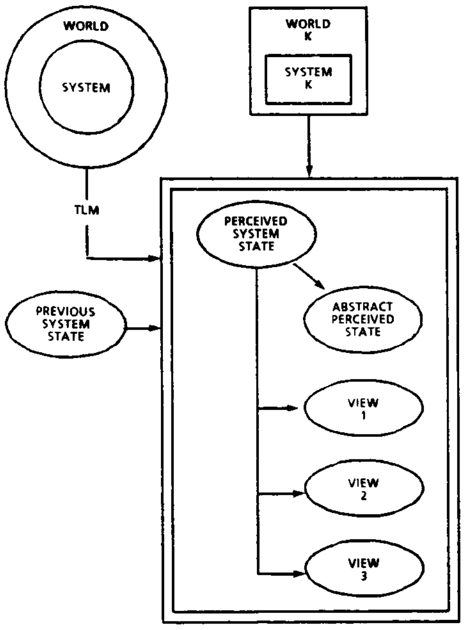


Figure 1. State Determination

d. **Perceived System State:** Current state of the world as perceived and derived from previous state, current observations, and system/world knowledge.

e. **Abstract Perceived State:** Abstractions of perceived state derived from previous state, current observations, and system/world knowledge.

f. **Views:** Other viewpoints or ways of looking at system state derived from previous state, current observations, and system/world knowledge.

Situation Assessment: Situation assessment assesses how well the current system state provides desired system capabilities. The key elements required to support this process are shown in Figure 2. They include:

a. **Perceived System State:** From State Determination.

b. **World K. System K:** Detailed knowledge about the system function, design, and behavior in the world.

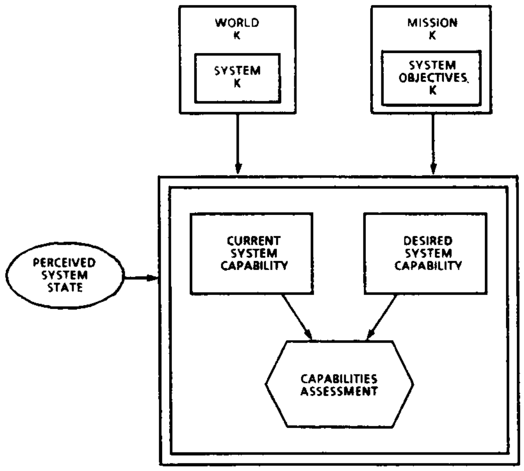


Figure 2. Situation Assessment

c. **Mission K, System Objectives K: Detailed knowledge about the mission and the system objectives that support various mission needs.**

d. **Measure Methods/Criteria K:** Knowledge about how to measure and assess how well a system state provides intended system capabilities.

e. **Current System Capability:** World K, System K, Mission K, and System Objectives K are used to determine what current system capability is provided from the perceived system state.

f. **Desired System Capability:** Mission K and System Objectives K are used to determine what mission and system capabilities are needed/desired for acceptable mission performance.

g. **Capabilities Assessment:** Measure Methods/Criteria K is used to assess how close current system capability is to desired system capability. In particular, what excesses or shortfalls exist between the two capabilities.

Problems Construction: Problems construction (see Figure 3) is a focusing activity that translates the results from situation assessment into a realistic desired functionality that takes into consideration constraints from current system status and knowledge about system behavior. Causal pathways are searched to identify concepts and behavior that are most likely involved in creating the current problems or in achieving desired functionality.

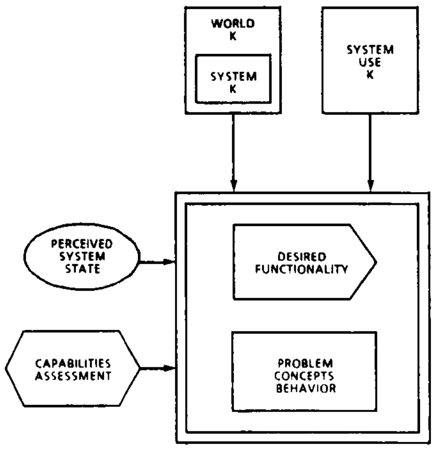


Figure 3. Problems Construction

Goals Construction: Goals construction (see Figure 4) focuses on problem concepts/behavior from problems construction and desired functionality to generate a desired system state. To accomplish this, we can follow state transitions for problem concepts to find behavior that can better achieve the desired functionality. Higher level knowledge of the relationships between concept states and system functionality will be necessary to help focus the search on paths that are most fruitful. Criteria for evaluating and comparing the functionality of different states will be used to decide on an acceptable desired state. Operator interaction may be integrated to assist in focusing resources.

Plans Construction: Plans construction (see Figure 5) involves finding recommended actions that will move the current system state toward the desired system state provided by goals construction. Transition conditions and causal pathways provide the information necessary to find these actions as well as to simulate forward in time to determine expected results of recommended actions including any potentially harmful effects. Criteria for evaluating and comparing the benefits and detriments of alternative actions will be used to rank various options for presentation to the decision maker. Operator interaction may be integrated to assist in focusing planning activities.

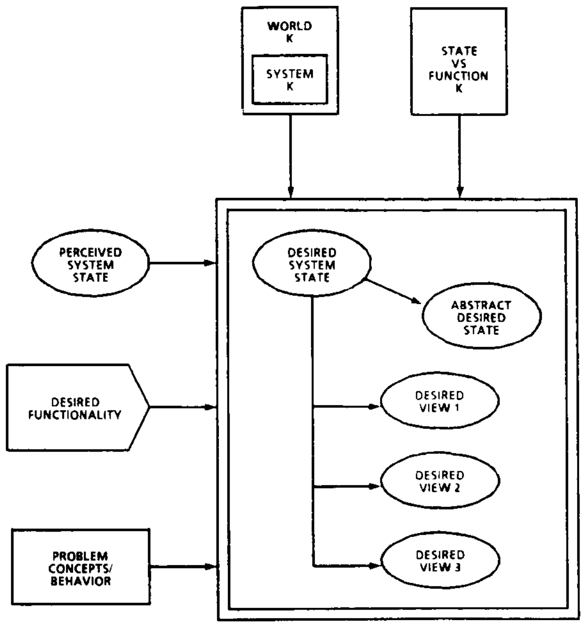


Figure 4. Goals Construction

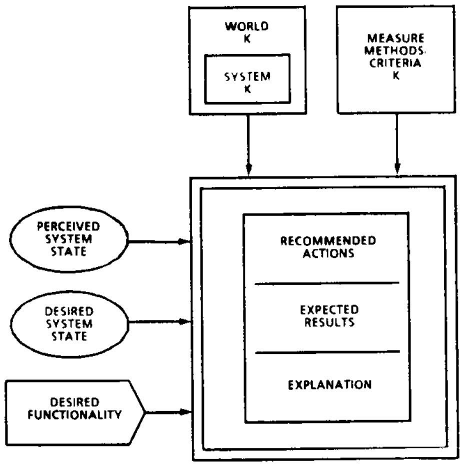


Figure 5. Plans Construction

Feedback Between Components

In the previous discussion, for simplicity, we neglected the feedback paths between components and the various levels at which each component might be working. See Figure 6 for a more complete process flow. Complete state determination for a space system is a large task requiring extensive computing resources and time. It is much more practical to maintain a small critical subset of current state information, and to enlarge the scope and depth when necessary to support other problem-solving component needs. Similarly, situation assessment will normally operate at a high level until something unusual or anomalous happens that triggers a need for increased depth and detail. This trigger could come from the high level assessment or from something required by problems construction. Problems construction, in turn, may need to expand or contract its focus as it performs its own job, or due to additional needs from goals construction or plans construction. Goals construction, also, may be influenced by what plans construction is able to achieve.

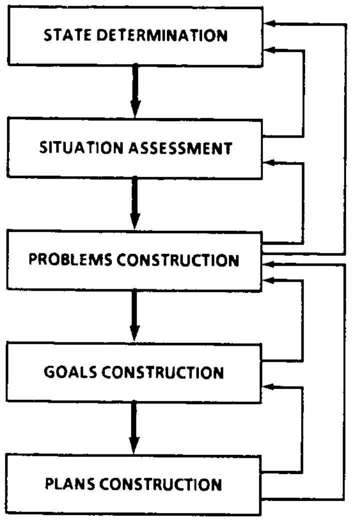


Figure 6. Decision Process Flow

CONCLUSION

Problem solving in the space operations domain is a very complex process. Building intelligent decision support systems that emulate human problem-solving functions and processes offer many difficult challenges, but they are challenges that can and must be solved to allow transition to the type of operations support and autonomous control currently envisioned for future space programs. Understanding the problem is the first step. Developing innovative approaches for knowledge representation, reasoning, and reasoning control that encompass the full breadth of problem-solving component needs in an incremental yet flexible manner is the next challenge. Extensive work is in progress to accomplish this. General solutions may be a long way off, but we expect that a variety of intelligent decision aids that augment and support key space operations activities will be developed in the next few years and that these will be the starting point for more comprehensive capabilities and approaches that are required for complex, sophisticated problem-solving support.

ANALYZYING SPACECRAFT CONFIGURATIONS THROUGH SPECIALIZATION AND DEFAULT REASONING

(Paper not provided by publication date.)

NASA USAF PROGRAMMATIC OVERVIEW

(No papers presented at this session)

MISSION PLANNING AND SCHEDULING

PRECEDING PAGE BLANK NOT FILMED

N91-20673

USER INTERFACE ISSUES IN SUPPORTING HUMAN-COMPUTER INTEGRATED SCHEDULING

Lynne P. Cooper
Eric W. Biefeld
Jet Propulsion Laboratory
California Institute of Technology
Mail Stop 301-440
4800 Oak Grove Drive
Pasadena, CA 91109
(818) 354-3252

ABSTRACT

A major problem in designing user interfaces for scheduling systems is one of allowing the human to become an integral part of the system. The human role in scheduling extends beyond the simple tasks of providing the input and accepting the output. Because of the inherent intractability of most real-world scheduling problems, intelligence must be incorporated into the scheduling process in order to reach an acceptable solution in a reasonable amount of time. Artificial Intelligence research has concentrated on identifying algorithms and heuristics for this purpose. However, interfaces which allow the scheduler to take advantage of human intelligence and allow the user insight into and influence over the planning process are also needed.

Enhanced interfaces support transitioning to a human-computer integrated mode of scheduling. This paper explores the user interface problems encountered with the Operations Mission Planner project at the Jet Propulsion Laboratory. OMP uses a unique iterative approach to planning which places additional requirements on the user interface, particularly to support system development and maintenance. These requirements are necessary to support the concepts of heuristically controlled search, in-progress assessment, and iterative refinement of the schedule. This paper presents the techniques used to address the OMP interface needs.

BACKGROUND

The Operations Mission Planner (OMP) is a multi-year research project currently in its third year. The goal of this project is to use Artificial Intelligence techniques to create an intelligent, automated planning and scheduling system. The need for advanced user interface capabilities to support this goal has been recognized. In addition to providing the general user with a means of interfacing with OMP, the user interface must also incorporate advanced features which support the special needs of system developers and maintainers [Becker 1987]. These include facilities to assist in the identification and development of heuristics, in debugging the planning logic, and in evaluating the quality of the schedule produced. The following sections identify problems in user interface design which have surfaced due to research currently underway on the OMP project along with the techniques being used to address those problems.

GENERAL PROBLEM DESCRIPTION

The interfaces to existing planning and scheduling systems range from those which provide only the final results of the scheduling process (e.g. Deviser [Vere 1983], RALPH [Webb and Yates 1987]) to those which provide incremental results during execution (e.g. PLAN-IT [Biefeld 1986], OMP-I). However, the information and rules that the planner used to reach the decisions are embedded deep in the planning process and are generally unavailable during execution. A functional user interface must provide a means of looking "behind the scenes" during actual scheduling in order to enable the user to understand the motivations for performing particular scheduling actions.

Current modalities for in-progress interaction limit the user to performing direct scheduling tasks (e.g. moving a task, deleting a task, specifying a task breakdown, or choosing from a program-generated set of predefined control options) during an actual scheduling run. For example, the interactive mode for OPIS system [Smith 1988], consists of several opportunities during the control cycle for the user to pick from a list of options. The user is unable to make small, real-time changes to the scheduling heuristics and is therefore unable to directly influence how the scheduler makes future, automated decisions without editing the code.

When the user makes a direct change to the schedule, the automated scheduler has no insight into the user's decision making process. The scheduler is left without a means of interpreting the significance of the user's action. If a user moves a task to a specific location, is the user indicating a preference or an absolute? Under what circumstances, if any, can the scheduler move that task?

Real-time evaluation of the quality of the schedule by the user requires advanced interface techniques. If the user only sees the results of the scheduling efforts up to a given point, he has only a limited feel for how the schedule is progressing. Have problem areas been identified and resolved? Has the schedule really improved -- or is it at an impasse? Is the schedule "good enough"?

The design philosophy for many user interfaces is to present the user with as much information as possible on the status of the system. Such interfaces often cause the user to suffer from data overload. Too many tasks, squeezed into too little space, using a representation which is difficult to interpret for complex problem domains, results in inefficient interfaces [Schneiderman 1987]. Incorporating additional modalities, without special consideration for form and functionality, results in unusable interface designs. Advanced techniques which filter user information to support only the function the user is performing, and which reduce the clutter on the screen without reducing information content, are vital in addressing the problem of data overload.

These problems have surfaced as a result of research currently being performed at the Jet Propulsion Laboratory in the areas of planning and scheduling. While the focus of the research has been on automating the scheduling process, it has become quite evident that improved methods of user interaction with the system are essential for system development and are highly desirable for acceptance of any resulting system by the user community. Because the interface issues are tightly coupled to the planning and scheduling research, it is important to understand some of the unique underlying concepts of the Operations Mission Planner.

PROBLEM DOMAIN

The Operations Mission Planner problem is one of resource allocation in a highly over-subscribed, under-constrained domain. There are three main areas of research in OMP: iterative planning, multiple control heuristics, and chronologies [Atkinson, et.al. 1988].

OMP is an iterative planner which progresses by making a series of passes over the schedule [Biefeld and Cooper 1988a]. Each pass further refines the schedule by performing a deeper, but more narrowly focused search. The purpose of these iterations is to use the information gathered during previous passes to guide the current pass. This information is kept in a variety of data objects, referred to collectively as chronologies. The chronologies are used by various heuristics which assess the state of the schedule, control the focus of the schedule and perform scheduling actions [Biefeld and Cooper 1988b].

OMP iterates through several phases to complete a schedule. Each phase has a specific goal, focus, and associated heuristics, and consists of several passes through the schedule. OMP first uses simple and very fast heuristics to load the schedule. In the next phase, it focuses on identifying resource bottlenecks. Once these bottlenecks have been identified, OMP switches to more powerful scheduling heuristics to resolve the conflicts existing in the bottleneck regions. The final phase of plan generation consists of "optimizing" the schedule. Since the system is so greatly over-subscribed, OMP uses this phase to maximize the number of tasks performed by the schedule.

OMP must also react in real-time to events occurring during schedule execution. An additional phase, the Event Handler, is responsible for initiating replanning activities based on its assessed impact of the event on the schedule. Since OMP is an iterative system, the Event Handler's primary function is identifying in which of the generation phases to reinitiate planning. Information gathered throughout the scheduling process remains available for the scheduler to use. The reinitiation process depends upon the severity of the event (e.g. simple task insertion vs. recovery from major resource failure) [Biefeld and Cooper 1990].

Each phase has different heuristics associated with it. These heuristics control the availability of scheduling actions, the basis for choosing between scheduling options, assessment techniques, guidelines for conducting search, and control mechanisms for identifying and progressing to the next phase.

Each major research area of OMP highlights pertinent user interface problems. How do we depict the iterative process to the users? The user must be allowed to interact with the heuristics to guide the scheduling process. How do we enable the system to interpret human intervention in the scheduling process? System developers and maintainers are responsible for identifying, testing, and incorporating new heuristics. How do we provide insight into the development of chronologies and the scheduling processes so the user can formulate new heuristics? These functionalities must fit into the user interface without overwhelming the user.

TECHNIQUES

Both the automated and human portions of a system such as OMP must be considered in defining the operational system [Potosnak 1987]. The OMP functional analysis identified the proposed breakdown of tasks between the automated and human segments. The automated segment is responsible for the process of scheduling, while the user is responsible for monitoring the scheduling process and for improving or creating new aspects of the system (e.g. heuristics). The automated scheduler should be able to develop, assess, and modify the schedule without the benefit of any additional input from the user. It must, however, be able to incorporate user direction when provided.

Since the user is responsible for identifying new heuristics and scheduling algorithms, he is ultimately responsible for assessing the quality of the schedule, and monitoring its execution. With the appropriate tools, the user can also play a vital role in identifying problems during the scheduling process, providing guidance, and directly manipulating the schedule.

Chronologies

In order to identify new heuristics and scheduling algorithms, the user must have insight into the iterative planning process. In order to do this, we must first provide insight into the development of chronologies. OMP's overall planning paradigm is based on empirical analysis of expert human schedulers [Biefeld 1986]. Initially, heuristics were developed to emulate the types of behavior exhibited by these experts. Additional heuristics have been discovered by watching OMP execute and focusing on specific parameters of potential importance. Observations of changes in the schedule representation as the scheduler progresses and off-line analysis of OMP performance for test cases revealed heuristics which decide how to configure resources, make a task-resource assignment, and determine the area of the schedule to work on next.

While initial analysis has proved useful, the level of complexity OMP will entail requires the use of tools which will make the observation, analysis, and synthesis tasks easier and more efficient to perform. Rather than looking at chronologies after the fact, we need to observe them as they are being built. The best method for doing this depends on the specific chronology, but methods which allow the integration of human abstract pattern recognition are essential.

The first area to warrant development of a special chronology interface was that of bottleneck identification. Currently, bottleneck regions are identified by performing a simple analysis of changes in the number of conflicts on the schedule resulting from scheduling activity on a specific schedule segment. The user interface supports more sophisticated analysis by: 1) monitoring specified chronology parameters, 2) performing trend analysis to indicate how the parameters are changing, and 3) presenting this information to the user in alternative formats.

Heuristics and Guidance

The OMP user interface must enable the user to interact with the scheduling heuristics. Our method of accomplishing this is through a real time edit capability. While the scheduler is operating, the user will be able to interrupt the scheduler to modify parameters associated with the control heuristics. This ability to "tweak" the system will provide greater control over the system and will serve as a test and evaluation aid.

This approach, however, has limitations. The heuristics must be defined in such a manner that their parameters are easily accessible and special safeguards must be incorporated to avoid causing system errors [Arens 1988]. The user must be provided with an understanding of how the heuristics work and the significance of the parameters in order to make meaningful changes. For an operational system, this places a heavy burden upon the user interface, and requires additional effort in defining heuristics. For the development phase, however, this overhead can be reduced.

Human guidance to the system can range from focusing the efforts of the scheduler on a particular segment of the schedule, to changing the pool of allowable heuristics for a scheduling pass, to providing specific instructions for a given task which differ from those originally provided (e.g. relaxing or adding constraints). At their most basic level, these forms of guidance can be thought of as editing existing definitions of tasks, resources, and heuristics. However, these editing tasks apply to a combination of data objects (tasks, resources) and processes (heuristics). Therefore, there is an overhead cost in the object representations associated with providing these editing features.

The scheduler must be able to interpret the relevance of a real-time edit to the original description of the object [Seeley 1987]. For example, if a user specifies that a task is to be assigned to a specific resource, is the scheduler allowed to disregard that assignment? If it does and the user once again makes that specific assignment, should it take the user more seriously? How does the scheduler know and interpret the level of preference? How do we help the user to provide this information?

In order to implement this edit feature, a temporary specification, or overlay, of an object description is used. Since OMP uses objects to represent tasks and resources, consistency must be maintained and inheritance features addressed when making temporary changes to an instantiation of an object type [Brachman 1985]. This overlay structure allows the system to operate using a modified description of an object, but does not remove the original specification which may be needed in later planning phases. The system can thus discard the overlay when it is no longer needed, does not have the overhead associated with creating and maintaining an overlay unless one is needed, and retains the original information during the scheduling pass affected by the overlay. A relative preference scale which indicates the importance of a given user action, as well as methods for manipulating such a scale by both the user and the scheduler, are planned.

Assessment of the Schedule

Assessment of the schedule in progress is an important aspect of the iterative planning process. In order to identify when to progress to the next phase, the scheduler needs to assess the effectiveness of continuing in the current phase. Such indicators as level of effort expended without additional conflict resolution, the appearance of cycles in the scheduling actions, or the need to perform a substantial amount of deletion to resolve conflicts will be used to identify when to progress to the next phase. A more difficult problem, however, is assessing the quality of the developed schedule.

In our problem domain, determining the quality, or "goodness", of the schedule is exacerbated by a lack of specific metrics upon which to base an assessment. The OMP problem domain, for example, is highly oversubscribed. Therefore, there is no schedule which can perform all of the requested tasks. The tasks themselves follow a strict and absolute priority ranking, so the sheer number of tasks performed is not an effective metric. Nor is there a metric which relates numbers of tasks with different priorities. OMP requirements dictate minimization of the number of tasks not performed, but not at the expense of the higher priority tasks.

The development of evaluation criteria will be an important aspect of future OMP research. There are user interface features which will assist the user in developing these criteria. Statistical comparisons of the numbers and distributions of requested tasks vs. those actually incorporated into the schedule can be developed. Response times, resulting performance, and changes in configurations caused by responses to events can all be monitored and made available for off-line analysis. Incremental information in these areas will be available during the scheduling process. These methods do not solve the problem of determining the characteristics of a good schedule. Rather, they perform an information gathering function which can be used to develop that definition.

Data Overload

The amount of information available to the user at any given time in the scheduling process can quickly become overwhelming. Special care must be taken to present information in an efficient fashion which permits the user to easily interpret the data [Tufte 1983]. Graphics and icons have become a popular means of representing data, but an appropriate level of abstraction must be selected. In OMP, for example, the tasks, in their most simple representation, are in the form of Gantt Charts. Unfortunately, the capacity of the screen is rapidly exceeded due to the sheer number of tasks which must be displayed.

Various "rich coding" and filtering techniques are used to reduce the visual confusion but maintain the level of information presented. Rich coding algorithms allow data to be represented at different levels of resolution or abstraction so that the user has a more intuitive grasp of the information presented without being overwhelmed by its magnitude.

Intent driven display techniques can also be used to reduce data overload [Madni 1982]. The information the user sees is filtered based on the task being performed. The user is spared from searching through potentially large amounts of extraneous data. Intent driven displays can interact with rich coding algorithms by setting the level of abstraction/resolution that these algorithms provide.

CONCLUSION

The types of interface features described in this paper present a departure from the basic paradigm of the user as a monitor of the planning system, rather than a participant in the planning process. In order for planning technology to progress to the point where it can be effectively automated, the issues discussed in this paper must be addressed. While the user interface concepts are from the perspective of an integrated human-computer scheduling system, the user is, in essence, acting as a heuristic. Therefore, in order to automate those human-heuristic functions, they must first be identified and generalized. It is necessary to understand where, when, and how the human user can have a positive impact upon the scheduling process. Only then is it feasible to address advanced automation.

Several of the techniques discussed in this paper are now being used in the Operations Mission Planner. The existing OMP prototype incorporates the Initial Load, Resource Centered, and Time Centered Phases of the iterative planning process. The user interface is in a preliminary state and uses graphics to represent the state of the schedule as it is in progress. Insights into the planning process itself, as discussed in this paper, and advanced presentation techniques such as rich coding and intent driven displays are planned additions to the interface.

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, and sponsored by the Department of Defense under contract No. 81-2877.

REFERENCES

Arens 1988
Arens, Yigal, Lawrence Miller, Stuart Shapiro, Norman K. Sondheim
Automatic Construction of User-Interface Displays
Proceedings of AAAI88, Volume 2, pp. 808-812, 1988.

Atkinson, et.al. 1988
Atkinson, D., E. Biefeld, L. Cooper, L. Falcone, and G. Martin
Operations Mission Planner Annual Report, JPL Document D-5685.

Becker 1987
Becker, R.M. and William A.S. Buxton
Human-Computer Interaction, A Multidisciplinary Approach, Research Frontiers and Unsolved Problems
pp. 669-676, Morgan Kaufmann Publishers, 1987.

Biefeld 1986
Biefeld, Eric
PLAN-IT: Knowledge-Based Mission Sequencing
Proceedings of SPIE on Space Station Automation
pp. 126-130, October, 1986

Biefeld and Cooper 1988a
Biefeld, Eric, and Lynne Cooper
Replanning and Iterative Refinement in Mission Scheduling
Presented at: The Sixth Intelligence Community Artificial Intelligence Symposium October, 1988

Biefeld and Cooper 1988b
Biefeld, Eric, and Lynne Cooper
Scheduling with Chronology-Directed Search, November, 1988

Biefeld and Cooper 1990
Biefeld, Eric, and Lynne Cooper,
Operations Mission Planner: Final Report, JPL Publication 90-16, March 15, 1990.

Brachman 1985
Brachman, Ronald J.
"I Lied about the Trees" Or, Defaults and Definitions in Knowledge Representations
AI Magazine, Fall 1985

Madni 1982
Madni, Azad, M.G. Samet, and A.Freedy
A Trainable On-Line Model of the Human Operator in Information Acquisition Tasks.
IEEE Transactions of Systems, Man, and Cybernetics, Special Issue on Human Factors in Computer Management of Information for Decision Making, Vol SMC-12, No. 4, July/August 1982.

Potosnak 1987
Potosnak, Kathleen
Designing Ergonomic User-System Interfaces
Pp. 90-110, The Koffler Group, Santa Monica, CA 1987

Schneiderman 1987
Schneiderman, Ben
Designing the User Interface, Strategies for Effective Human-Computer Interaction
Addison-Wesley Publishing Company, 1987

Seeley 1987
Seeley, D.A.
An Object Oriented Reference Model for User Interface Design, Hypertext, and Collaboration. in *Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems*, edited by G. Salvendy, Elsevier Science Publishers, 1987.

Smith 1988
Smith, Stephen, and Lynne Cooper
Personal Conversation, December 1988.

Tufte 1983
Tufte, Edward R.
The Visual Display of Quantitative Information
Graphic Press, Cheshire, Connecticut, 1983.

Vere 1983
Vere, Steven
"Planning in Time: Windows and Durations for Activities and Goals"
IEEE Transactions on Machine Intelligence, No. 3, pp. 246-267, 1983.

Webb and Yates 1987
Webb, W. Allen and Gigi L. Yates
Interactive Graphics Applications for Allocating Resources for the NASA Deep Space Network, JPL Document D-4628 May, 1987

N91-20674

ARCHITECTURE FOR SPACECRAFT OPERATIONS PLANNING

William S. Davis
Boeing Computer Services
Artificial Intelligence Center
PO Box 240002, MS JA-74
Huntsville, AL 35824

ABSTRACT

The dynamic environment of working in space presents several challenges to planning space operations. One challenge is the heterogeneous nature of the operations to be performed, ranging from life science experimentation to vehicle assembly. Generating plans for such diverse tasks requires a system that exhibits many domain-independent characteristics. A second major challenge is that the performers of these operations (plan agents) are also heterogeneous, possessing varying skills and physical capabilities. In this regard, planning must be possible both separately from, and in consideration of, each agent's capabilities, whether crewmember or robot. Finally, operating in space encompasses unanticipated events. By definition, no pre-established (or "canned") plan can accommodate.such situations; hence a system is required which can dynamically plan and replan according to evolving knowledge.

We present such a system which generates plans for the dynamic environment of space operations. This system synthesizes plans by combining known operations under a set of physical, functional, and temporal constraints from various plan entities (agents, objects, tasks), which are modeled independently but combined in a flexible manner to suit dynamic planning needs. This independence allows the generation of a single plan source which can be compiled and applied to a variety of agents. Another result of this modular planning concept is the ability to generate different plans from the same instructions, according to the objects used in the plan. The architecture blends aspects of temporal logic, nonlinear planning, and object-oriented constraint modeling to achieve its flexibility. In our operations testbed, we have applied this system to the domain of IVA maintenance and repair aboard Space Station Freedom, planning operations for (1) a crewmember, generating English statements and interacting using speech synthesis and voice recognition systems; (2) a one-armed robot, generating robotic programming instructions for sequential actions; and (3) a graphical simulator, generating software commands for a NASA/Vanderbilt simulator modeling one- and two-armed robots.

1. Introduction

The domain of mission operations, as applied to spacecraft such as Space Station Freedom (SSF), is characterized by both long life cycles and a broad scope of activities. For example, Space Station Freedom is expected to perform orbital operations for approximately thirty years, during which it will serve a variety of both scientific and space exploration needs. Scientific activities vary in types of experimentation, including life sciences, materials processing, astronomy, automation and robotics, and other disciplines applying primarily to spacecraft operations. Space exploration includes the orbit-based activities of assembling, testing, servicing, launching, and recovering spacecraft (Lunar and Mars Transfer Vehicles) and the surface-based activities of navigation, experimentation, and outpost establishment [1]. In addition, routine activities (housekeeping, maintenance, repair) must be performed for each craft to maintain proper operating status of its components.

2. Planning Issues Associated with Spacecraft Operations

A recent NASA/OAST study has indicated a long-term research focus on intelligent agents to support automation for space exploration [2]. Planning for these intelligent agents, however, faces challenges created by properties inherent to the domain. First, the tasks associated with spacecraft operations are heterogeneous. As mentioned above, activities include experimental research, spacecraft assembly and checkout, maintenance, repair, and safety monitoring Each of these areas is a domain unto itself, thereby having its own set of methods and heuristics to reason about actions. Hence, it is clear that automated planning systems developed for spacecraft operations must possess a significant amount of domain-independence, and yet allow a modular combination of sophisticated constraints and heuristics in applying specific problem-solving strategies. In addition, with heterogeneous agents performing a heterogeneous set of tasks, a proper balance must be maintained between distributed and global reasoning.

Second, the agents involved in spacecraft operations are also heterogeneous. Each agent possesses a unique combination of physical and functional capabilities, emphasizing distinct skills. Crewmembers, for example, possess differing specialties (e.g., a flight officer, a telerobotic expert, a life-sciences payload specialist) while being similar in physical capabilities and general functionality. On the other hand, on-orbit robots differ greatly in physical capabilities (e.g., Flight Telerobotic Servicer, Special Purpose Dextrous Manipulator, spacecraft assembly cranes, IVA maintenance robots), and by nature address different functional needs.

Third, this environment is naturally dynamic. For example, Space Station Freedom's payload operations will be scheduled into contiguous 90-day increments. Every increment will see new objectives and priorities, while possibly maintaining continuing activities from previous increments. Additionally, there will always be the potential for immediate-term adjustment of priorities to service needs such as emergencies or windows of unexpected opportunity. Such changes in activity schedules call for a strong emphasis on both replanning and monitoring of plan execution. Another dynamic aspect of the domain is evolution. The environment about which the reasoning occurs may evolve as systems are upgraded or reconfigured, thus changing their associated operations tasks. Crew agents will likely evolve as they expand and improve their skill sets through increased space habitation. Robotic agents will similarly evolve as changes in technology produce enhanced capabilities. These continuous changes in agents, their environment, and their activities suggest a requirement for automated planners that are highly flexible and extensible.

3. Strategies Addressing Planning Issues

This section discusses aspects of the architecture of on-going investigations at Boeing's Defense and Space Group, Huntsville Division, which address some of the issues highlighted above. Figure 1 depicts an abstraction of the architecture from which this discussion draws. The intent of this research is to provide a foundation which supports multiple aspects of the automated planning problem, including representation, reasoning schemes, and verification.

The issue of heterogeneous agents performing heterogeneous tasks is primarily being addressed with a methodology known as "agent-independent planning" [3]. Agent-independent planning generates activity plans using only the constraints associated with the tasks to be performed and the objects in the environment. Once a plan is generated, an agent is selected for whom the plan is validated, and instructions particular to that agent are translated from the original plan representation (see figure 2). This separation of constraints supports a planning system that is robust in allowing the evolution of tasks independently from the evolution of agents. The primary representation for activity is based on the temporal-interval logic established by James Allen [4]. Networks of temporal intervals form a hierarchically abstracted plan space to describe complex tasks. The system's constraints, which are arranged in a class structure and are similar to the basic planning constraints in SIPE [5], are dynamically grouped as combinations of tasks, objects, and agents are constructed to satisfy particular planning goals. The mechanism for plan generation is a combination of nonlinear and temporal planning techniques. While the very difficult problem of reasoning about multiple interacting agents is not currently addressed by this methodology, it does provide mechanisms to reason about agents with varying skills, and to determine validity among combinations of plans and agents.

The temporal logic employed by the planning system affords a rich scheme for representing activity within the complex domain. However, maintaining logical consistency among the temporal networks is computationally expensive. Of particular difficulty is dealing with the dynamic nature of spacecraft operations, where domain information is sometimes incomplete. This issue is being addressed by developing nonmonotonic capabilities into the temporal logic used by the planner. This provides the ability to retract temporal assertions made by previous inferences without disrupting previous propagations of constraints which remain valid. This capability allows more extensive use of temporal planning techniques, and in particular gives rise to replanning algorithms which also exploit this representation.

For automated planners to be successfully adopted into the potentially-hazardous domain of space, their reasoning methods must be verifiable to ensure safe and consistent operation. This includes not only verifying planner rationale, but also proper execution of the plans by the intelligent agents. Validation of plans during execution is being addressed by using techniques of model-based diagnostics to monitor and detect faults. Just as fault sensors in a physical system can be used to signal failures and characterize symptoms, an agent's sensors provide feedback which can be combined with plan causality structures and models of the environment to determine if and where a plan has failed. Fault detection and isolation algorithms provide a focus for replanning efforts to correct the plan error. Pre-execution plan validation is being addressed by investigating an explanation system which describes the rationale used in generating plans. This rationale is constructed from the plan's causal links, temporal relationships, and models of the domain environment (physical, functional, spatial, etc.). Explanation will be a key factor in integrating human agents with automated planning systems.

4. Directions for Research

In addition to the emphasis on intelligent agents, a recent OAST workshop identified both multi-agent reasoning and verification of automated functions as areas needing technology development [6]. While these areas encompass a broad spectrum of technology within automated planning, this section underscores a few that particularly address spacecraft operations issues.

Multi-agent reasoning, in this context, concerns not only the interaction and cooperation of agents but also the mere notion of multiple, heterogeneous agents available for a large variety of tasks. A critical factor to successful reasoning, then, is how well the domain is modeled. Planning for a complex environment requires better representations to deal with the complexity. Current logic and frame representations have afforded continuity in developing planning technology, but tools such as these should be extended to provide more encompassing representations. For example, the temporal interval logic has allowed significant advances in plan reasoning, but must be enhanced to incorporate complex domain issues such as nonmonotonicity. Spacecraft operations should be modeled in a manner that allows reasoning that can determine appropriate amounts of interaction among agents. Such models should address questions of how and when activity can be distributed, where should reasoning be performed, and how accurately must plans be developed.

Rich representational capability is needed to model not only plan behavior, but also the agents and the physical environment. Characteristics of agents should be modeled to facilitate applying agents to meet dynamic needs. The physical and functional models of agents should provide planning constraints not only with agent capabilities, but also with more abstract knowledge such as skill proficiencies, task preferences, and endurance. In terms of cooperating agents, the complex issue of an agent's knowledge or belief of other agents is compounded when certain assumptions can possess hazardous consequences. Representations of the physical environment should follow along the lines of model-based reasoning. Insight into physical component functionality will allow planners to match agent capabilities with object behavior to achieve complex goals.

Verification of automated functions, while difficult, is intensified when applied to the spacecraft operations domain, where there is usually little margin for error. It is critical not only for reliable performance of operations, but also for acceptability of automation among spacecraft personnel. The rationale used in deriving plans must be shown to be satisfactory in order for crewmembers to rely on automated planning. Verification of planning rationale should address multiple levels of resolution, depending on the accuracy desired and time needed to perform the verification. Knowledge-based, model-based, and explanation-based methods are viable emphasis areas for performing and reporting plan validation. Ensuring reliable operation also extends into plan execution and monitoring. Hence, replanning, reactive planning, plan diagnosis, etc. are all valid areas of attention. What will make them useful for spacecraft operations is their integration with complex representations, as described above, to perform more comprehensive reasoning about all aspects of the environment.

References

[1] *Report of the 90-Day Study on Human Exploration of the Moon and Mars*, Aaron Cohen, Study Director, NASA, November 1989.

[2] *Review of Human Exploration Program Technology Requirements*, NASA OAST presentation, September 6, 1989.

[3] *Agent-Independent Task Planning*, W.S. Davis, Proceedings of Fifth Conference on Artificial Intelligence for Space Applications, Huntsville, Alabama, May 1990.

[4] *Maintaining Knowledge About Temporal Intervals*", J.F. Allen, Communications of the ACM, vol. 26, pp. 832-843, 1983.

[5] Practical Planning: Extending the Classical AI Planning Paradigm, D.E. Wilkins, Morgan Kaufmann Publishers, San Mateo, California, 1988.

[6] *Technology for Space Station Evolution*, NASA/OAST Workshop, Dallas, Texas, January 1990.

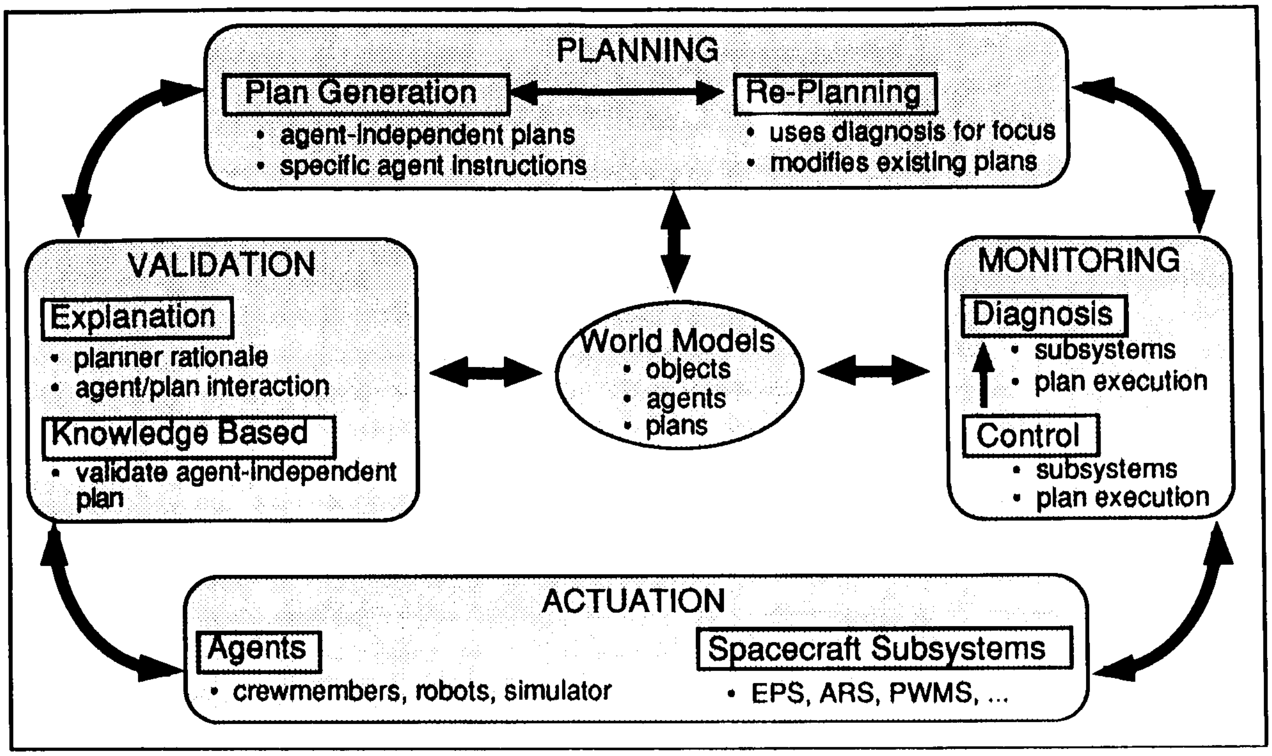


Figure 1: High-level Architecture for Operations Planning

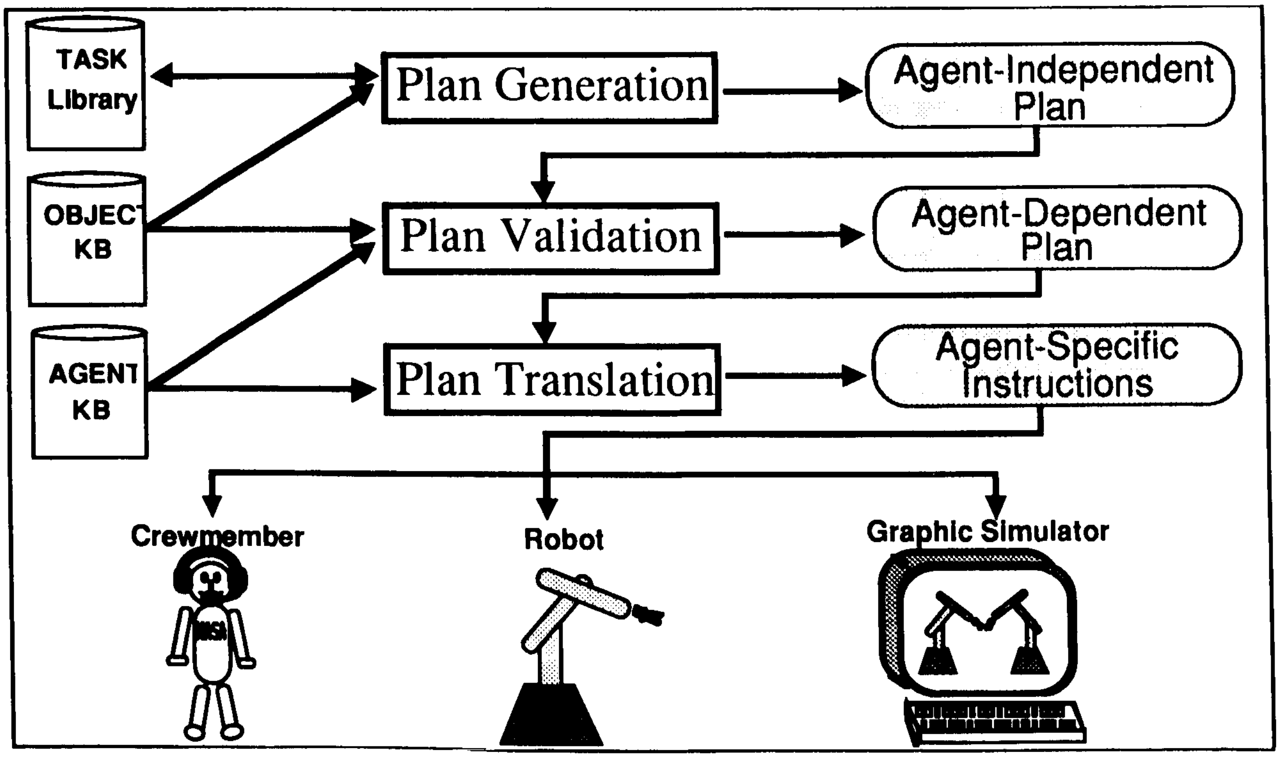


Figure 2: Agent-Independent Planning Process

THE IMPLEMENTATION OF COMPASS

Barry R. Fox
McDonnell Douglas Space Systems Company

(Paper not provided by publication date.)

N91-20675

The Range Scheduling Aid

Eliezer M. Halbfinger
The MITRE Corporation
Burlington Road
Bedford, MA 01730
emhalb@mbunix.mitre.org

Barry D. Smith
The MITRE Corporation
Burlington Road
Bedford, MA 01730
bds@mbunix.mitre.org

ABSTRACT

The Air Force Space Command schedules telemetry, tracking and control (TT&C) activities across the Air Force Satellite Control Network (AFSCN). This is known as range scheduling. The Range Scheduling Aid (RSA) is a rapid prototype developed by the MITRE Corporation combining a user-friendly, portable, graphical interface with a sophisticated object-oriented database.

The Range Scheduling Aid has been implemented as a set of five modules: the object-oriented database, the constraint-based analytics, the user interface, the multi-user blackboard system, and a dispatcher through which all modules communicate.

The objects in the object-oriented database have a one-to-one correspondence with the objects in the real world. They include satellites, tracking stations, pieces of equipment, requests for service and scheduled tasks.

The analytical capabilities of the Range Scheduling Aid include conflict identification, conflict explanation, and conflict resolution. It also has three error checking routines: time checking, location checking, and visibility checking.

The user interface to the RSA prototype is an electronic clone of the current paper chart. The user interface functions as a query/manipulation language for the database. By pointing at an icon with the mouse, all relevant information is displayed across the bottom of the screen. The user can choose operations from context-sensitive menus or drag an icon to a new location with a single mouse button.

To support multiple users and connections to the outside world, the RSA object-oriented database is populated by a subset of data from a commercial RDBMS. Modifications to an object on screen are posted via a modified blackboard architecture to the RDBMS and to all users affected by the change. Periodically the code requests its messages and updates its local database.

INTRODUCTION

The MITRE Corporation is developing an aid for the scheduling of resources in the Air Force Satellite Control Network (AFSCN). The Range Scheduling Aid (RSA), prototyped through frequent contact with the Air Force schedulers, maintains the currently employed scheduling techniques and efficiencies that have developed over the past twenty-five years, while adding the analytical and database capabilities of a knowledge-based system.

The Current Method of Range Scheduling

The Air Force Space Command (AFSPACECOM) schedules telemetry, tracking and command (TT&C) activities across the AFSCN. The resources of the AFSCN, called the "Range", include 15 remote tracking stations (RTSs) and 55 DOD satellites owned by 10 Mission Control Complexes (MCCs). Range scheduling, as this activity is known, is currently done manually on a three foot wide paper chart. The "acquisition chart" represents time across the x-axis and has horizontal bands delimiting the different tracking stations at which requests for service (satellite contacts as well as maintenance and testing) may be scheduled. The schedule of supports is created by placing one-quarter inch wide pieces of adhesive tape of the proper length at a specific time and RTS on the paper chart. The tapes themselves are color and pattern coded to represent specific satellites and their owning MCCs. An 84 foot long segment of the chart represents one week of the schedule to a granularity of one minute.

The schedulers receive paper forms of requests for satellite contacts, called "program action plans" (PAPs), from each of the MCCs as well as other requests for systems maintenance and testing. The schedulers must consolidate all of the requests into one conflict-free schedule. The schedule is created obeying the constraints of the request, including the request time window, the limited line-of-sight visibility of a satellite to an RTS, the specific set of equipment required for a satellite contact at an RTS, and the availability of the equipment.

The scheduling method, although optimized through its long period in use, is limited. Much of the data necessary to create a schedule, including the equipment available at each site, the equipment required for a satellite contact, the satellites' visibilities from the tracking stations and many of the PAPs, is available to the schedulers only in hardcopy form. Much training time is spent becoming proficient with the volumes of data. Scheduling depends on the paper chart and can therefore only be done at a single location. The schedule as finalized on the chart must then be manually typed into a computer system to be sent to the RTSs and MCCs. Understandably, many operator errors are initiated in the data input step. As the number of resources and requests in the domain increases, the maximum capacity for manual scheduling is being approached.

Range Scheduling Aid Architecture

The Range Scheduling Aid prototype has been implemented as a set of five modules: the object-oriented database, the constraint-based analytics, the user interface, the multi-user blackboard system, and a dispatcher through which all modules communicate (see figure 1). The majority of code in the RSA is written in Common LISP with only a few graphics functions dependent on the LISP implementation. The single user version of the system, which was the main focus of development during the first two and one-half years of the effort, runs on Symbolics machines, Sun workstations, TI MicroExplorers and Apple Macintosh IIs. The blackboard module for the multi-user version and the interface between the single RSA nodes and the blackboard are written in C. The multi-user version is currently running on Sun workstations.

The Range Scheduling Aid (RSA) combines a user-friendly, portable, graphical interface with a sophisticated object-oriented database. The RSA maintains the appearance and functionality of the paper chart while adding many advanced capabilities to assist the scheduler. A sophisticated object oriented database that represents the real world entities of the Range can be queried directly through the user interface. Analytical routines identify errors and conflicts in the schedule and also generate possible alternative solutions to create a completely conflict-free schedule. The multi-user module allows multiple users to concurrently schedule on separate workstations using data archived and updated to a single COTS relational database management system. The RSA system also allows a schedule to be electronically disseminated to the MCCs and RTSs through the COTS RDBMS rather than requiring error-prone manual input of the data. As up to fifty percent of the tasks are changed in the 24 hours prior to the mission, changes to the schedule in real time can also be done more efficiently and the adjustments disseminated more quickly.

THE OBJECT-ORIENTED DATABASE

The entities in the object-oriented database have a one-to-one correspondence with the objects in the real world. They include satellites and non-flight activities, tracking stations, pieces of equipment, requests for service and scheduled tasks. The database has multiple points of access to facilitate efficient retrieval of data using various identifiers.

A request for service specifies the satellite (or other activity) involved, a list of possible RTSs that will satisfy the request, as well as the time window during which the contact should occur and a preferred start and stop time. Any special equipment needed is also noted. When a request is scheduled, the scheduled task, set for a specific time, duration and location has pointers to the satellite, the RTS, the visibilities for the satellite at the chosen RTS, the equipment specified for the contact, and any other task which is in conflict with it. The

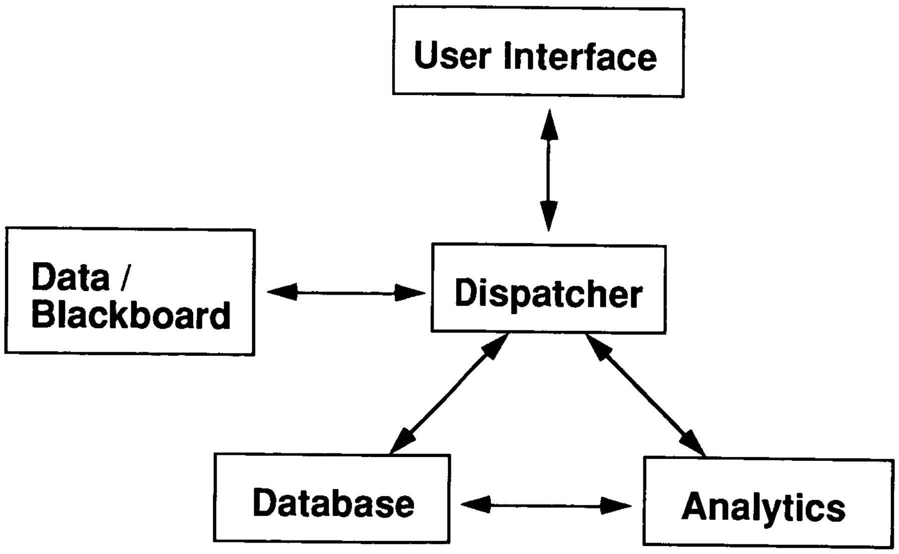


Figure 1: The RSA single node architecture

equipment necessary for the support of each satellite at all of the different tracking stations is stored in the environment table.

Queries to the database are performed exclusively through the mouse-driven user interface. By simply highlighting an icon on screen or bringing up a popup menu all pertinent data is displayed on screen. Data manipulation is also handled through the graphics by either dragging an icon to a different time and RTS or by accessing the menu for an on-screen object.

Analytical Functions

The analytical capabilities of the Range Scheduling Aid include conflict identification, conflict explanation, and conflict resolution. Scheduled requests, also called tasks or supports, are in conflict when they require the same resources for an overlapping period of time. There are three varieties of conflicts: equipment, turn-around-time (TAT), and range conflicts.

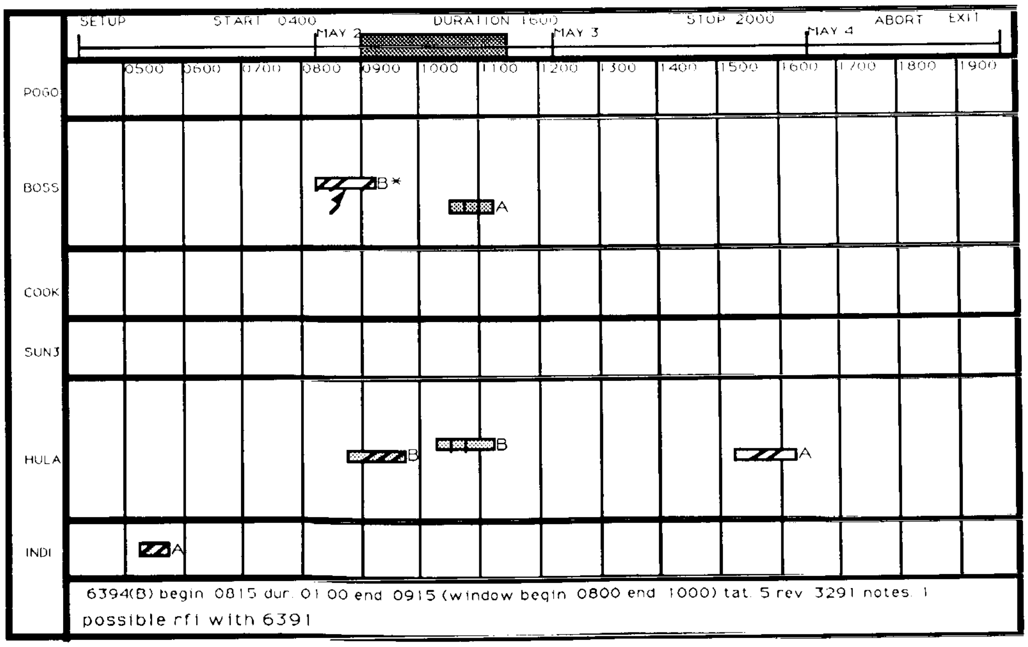
An equipment conflict occurs when two tasks at the same RTS are allocated the use of a given piece of equipment at the same time. A turn-around-time conflict is a similar overlap when the time required to set up for a support is subtracted from its begin time. A range conflict tests globally available equipment that can simultaneously support a finite number of tasks across the Range. The conflict identification routine is run whenever a change is made to a support; this may be a change in time, RTS, or equipment.

The conflict resolution procedure will find all of the possible conflict-free options for locating a scheduled task. The user is shown a pop up menu and can choose one option to relocate the task to a new position. When the conflict resolution algorithm is run for a given task, the task is first deleted from the database. A temporary task is created that has a length of the complete time window of the original request of that task. This task is placed at all possible RTSs to identify the other tasks which may possibly lead to a conflict. With this set of pertinent tasks, the algorithm searches for a time gap large enough to accommodate the length of the original task including its turn-around-time. Each of these solutions is returned and the original task is replaced in the database.

The RSA also has three error checking routines: time checking, location checking, and visibility checking. The time of the support is checked with the time window specified in the request for service and an error is flagged if the task is not completely within the window. The RTS at which the task is scheduled is compared with the request's list of possible RTSs for this task. The time of the task for a flight task is compared with the satellite's visibility data to ensure that the satellite has line-of-sight visibility with the RTS throughout the time period when it has been scheduled.

THE USER INTERFACE

The user interface to the RSA prototype is an electronic clone of the current paper chart (see figure 2). The user interface also functions as a query/manipulation language for the underlying database. The user can choose operations from context-sensitive menus or drag an icon to a new location with a single mouse button. The RSA



6394(B) begin 0815 dur 01 00 end 0915 (window begin 0800 end 1000) tat 5 rev 3291 notes 1
possible rfi with 6391

Figure 2: The User Interface

includes its own generic window system supporting several types of pop up menus, color bitmap manipulation and mouse-tracking across panes.

The layout of the screen has been adapted to allow the clear representation of the chart on a commercially available color monitor. Across the top of the screen is a pane with an adjustable timeline marking the time section displayed in the main screen area. The displayed work area can be adjusted by clicking the mouse in the time-pane and then dragging the shaded region to the desired time period. Any length period of time can be selected and the display can represent the schedule clearly with a typical data set for a 24 hour period. Discussions with the schedulers have revealed that although they currently have the ability with the paper chart to stand back and see the complete view of a whole week's schedule, they typically work on periods of twelve hours or less. The horizontal panes on the display coinciding with the available RTSs are also filtered in the RSA through a popup menu. As it is not feasible to show all 15 RTSs with ample work space for each, there is a menu for the user to selectively set the relative heights of each RTS to be displayed. RTSs can be completely hidden from view by setting the relative height to zero. In this way a scheduler focuses on a smaller portion of the scheduling problem but can easily change the display to refer to other data.

On the paper chart, a change in the schedule is accomplished by physically repositioning the tape. Notations concerning a task are written down in pencil on the paper near the placed tape. Changing the position of a tape requires erasing and rewriting its accompanying notes. The RSA defines these tapes as graphical objects. By pointing at an icon with the mouse, all relevant information is displayed across the bottom of the screen. In moving an icon by simply holding down the mouse button and dragging it, all notations for that icon are moved with it. Scheduling a support in error will display a notification menu and a red marker that remains next to the tape. A task in conflict is designated on screen with a pink bar through the patterned icon similar to the method used by the schedulers on the paper chart. To see the explanation of a conflict, a user chooses an option from an icon's menu and the causes of the conflict and the set of conflicting tasks (and equipment involved) are identified.

The RSA also includes icons for service requests, satellite visibilities and notes. Because of the volume of data, the RSA permits the scheduler to display and erase the icons that may not be pertinent to the schedulers current task. The data, however, remains in the database to perform the proper analytical functions.

MULTIPLE USERS - THE BLACKBOARD MODULE

To support multiple users and connections to the outside world, the RSA multi-user version includes the ability to instantiate several scheduling nodes from a single relational database and to manage the concurrency of data among multiple schedulers working on intersecting time periods (see figure 3). The RSA object-oriented database is populated by a subset of the range data from a commercial RDBMS. The central database is completely independent of

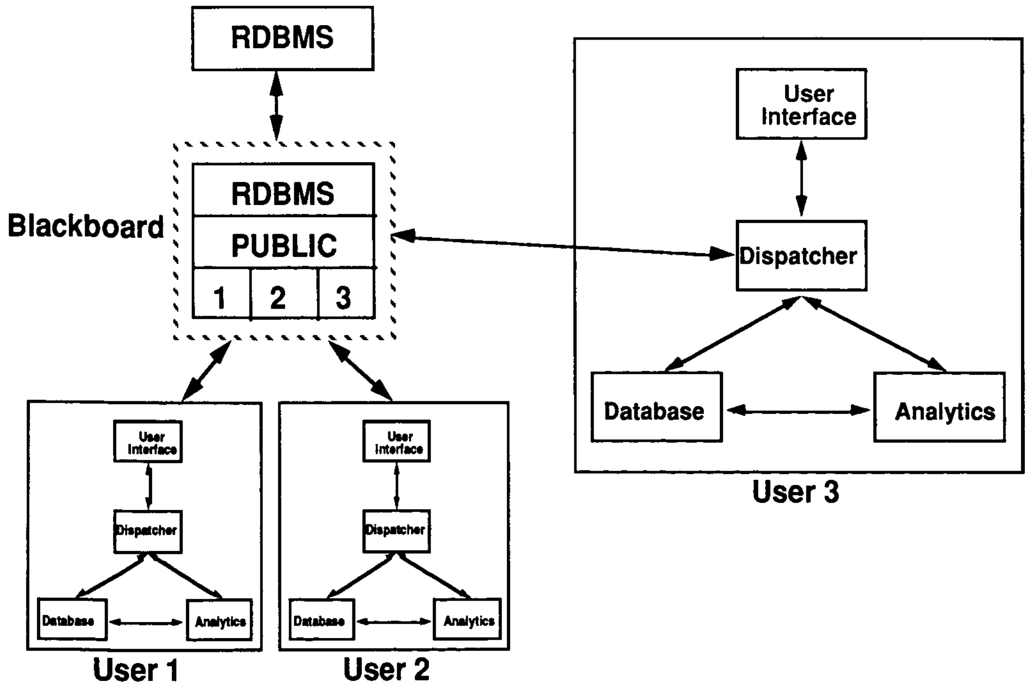


Figure 3: The Multi-User Architecture

the local object oriented databases and does not replace its use in any of the queries or manipulations done by the user. Each of the schedulers can therefore limit its object oriented database to a small portion of data and maximize performance while the complete volume of data remains in the disk-based RDBMS. Modifications to an object on screen (or in the local object-oriented database) are posted via a modified blackboard to the RDBMS and to all users affected by the change. Periodically, each workstation queries the blackboard and retrieves its queued up messages in order to update its local database.

The blackboard module itself has three components: an interface to the relational database management system, mailboxes for each of the active scheduler workstations, and a public area with data accessible to all of the scheduler nodes. The individual workstation passes the data for a message from LISP through a bidirectional stream to a background process. This process (written in C) is awakened and transmits the message across the network to the blackboard server using the SunOS Remote Procedure Call (RPC) interface. The server parses the message, and performs the necessary functions. If an update to the relational database is necessary, the data is formatted as required for accessing the database management system and the Standard Query Language (SQL) code is executed. The message, including any necessary data retrieved from the RDBMS, is then posted to the mailboxes of all other users interested in this message. The blackboard stores the time period for which each of the workstations is logged and checks whether each message pertains to the workstation. Messages concerning the locking of objects for update are posted to the public area. Before a scheduler is permitted to alter the data of an object, an exclusive lock must be successfully placed on the object. If another scheduler has locked that object, the modification is refused and the scheduler is notified with a popup menu.

CONCLUSION

The Range Scheduling Aid has been a rapid prototyping effort whose purpose is to elucidate and define suitable technology for enhancing the performance of the range schedulers. Designing a system to assist the schedulers in their task, utilizing their current techniques as well as enhancements enabled by an electronic environment, has created a continuously developing model that will serve as a standard for future range scheduling systems. The RSA system is easy to use, easily ported between platforms, fast and provides a set of tools for the scheduler that substantially increases his productivity.

Acknowledgements

This work has been funded by the US Air Force Systems Command, Electronic Systems Division at Hanscom AFB.

References

Hayes-Roth, Barbara, "A Blackboard Architecture for Control," Artificial Intelligence 26:3, June 1985.

Smith, Barry D., "A Description of the Range Scheduling Aid Database," MITRE Working Paper No. 28025, The MITRE Corporation, Bedford, MA, January 1989.

Smith, Stephen F., "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach," final report for AFOSR Contract No. F49620-85-C-0054 of the Electronics and Material Sciences Department of the Air Force, January, 1987.

N91-20676

Request Generation II Mission Planning and Scheduling

Darlene D. Greenhut
IBM-FSD
685 Citadel Drive East
Suite 400
Colorado Springs, Colorado
80919

Request Generation II (ReGe II) is a PC-based prototype knowledge based system intended to assist USAF personnel in planning and scheduling satellite operations for their Mission Control Complexes (MCC). It aids MCC personnel in producing weekly Program Action Plans (PAPs) for each of the satellite vehicles an MCC is responsible for monitoring and maintaining. The PAPs are input to the Resource Control Complex (RCC) which schedules all satellite support requests for usage of the network.

ReGe II is an IR&D project that combines the concepts of two previous projects, Request Generation (ReGe I) and MCC Scheduling Automation (MSA). The aim of ReGe II is to assist USAF personnel by evolving the task of vehicle planning and scheduling away from manual and repetitive data sorting with the use of automation with the intent of freeing the PA up to place emphasis on the more important aspect of the task: understanding the changing needs of the US owned satellites. This is particularly important in the current environment where satellites are increasing in number and complexity and staff is frequently turning over due to job rotation.

The future intent is to imbed this planning and scheduling capability within the future CCS-2000 architecture. This will provide an automated function for populating the network with maintenance requests and will provide the connectivity between the MCC operators and the existing configuration controlled mainframe databases.

Satellite Support Planning Process

Monitoring the health and performing maintenance on the US owned satellites is a responsibility divided among a number of Mission Control Centers (MCC). Within each MCC, a staff of Planner Analysts (PAs) are responsible for understanding the needs of the vehicles which the MCC controls. On a weekly basis Program Action Plans (PAP) are produced which detail support requests for the network resources needed to monitor, command and control with each vehicle. The PA´s are guided by three objectives when planning for vehicles:

o Requests for support need to satisfy vehicle requirements documented by Tests Operation Instructions (TOIs), Test Operation Orders (TOOs) and Memograms.

o Requests for resources should be combined where possible to reduce the burden on network resources.

o Requests should be as flexible as possible, allowing the largest time window within which a support needs to be scheduled.

Currently the planning process begins with the PA sitting down with paper listings describing vehicle events such as acquisition of the vehicle, vehicle sun eclipses or sun-vehicle-earth angles.

Next, the PA reviews documents describing the vehicle requirements (TOIs, TOOs and memograms).

Particular attention must be paid to requirements which may have changed since the last time the PA did planning for that specific vehicle [Figure 1].

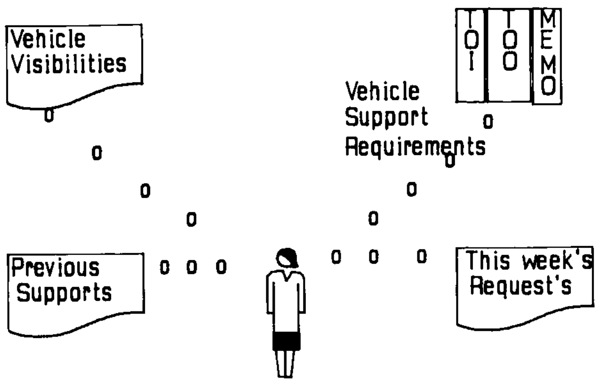


Figure 1:
Planner Analyst Tasks

The plan which the PAs develop to satisfy the vehicle requirements is recorded on paper and carried to a computer terminal, the data is entered into the system, and transferred to the Resource Control Center (RCC).

The RCC is responsible for scheduling the range resources based on the requests made by all MCCs. Conflicts are resolved via phone calls to individual MCCs. Once the range schedule is conflict free, the schedule is published.

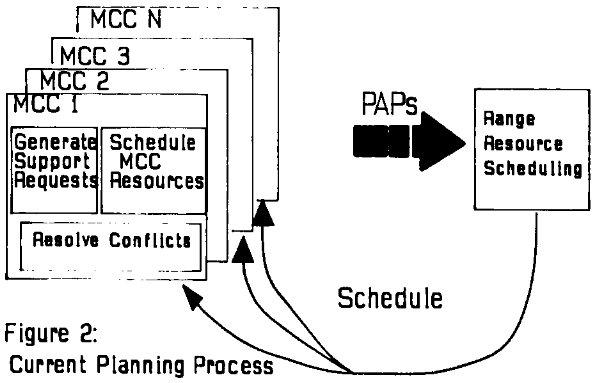


Figure 2:
Current Planning Process

The MCC plans will be readjusted to the new schedule and the next phase of vehicle planning can begin [Figure 2]. Until recently, PA tasks were performed by civilians with extended experience. These tasks are now being transitioned to DoD military personnel. Development of tools which will assist in training, performing the job and in turnover due to rotation become very important in this environment.

Much of the work that has been done in this arena has focused on the scheduling task within the RCC. IBM's work began with research on scheduling tools for the RCC led by Dr. Mansur Arbabi. During that research period, IBM won the contract which developed the RCC scheduling software and is currently on contract to maintain it.

Research done on scheduling within the RCC branched into MCC scheduling. A series of projects followed, each building on the lessons learned from the previous

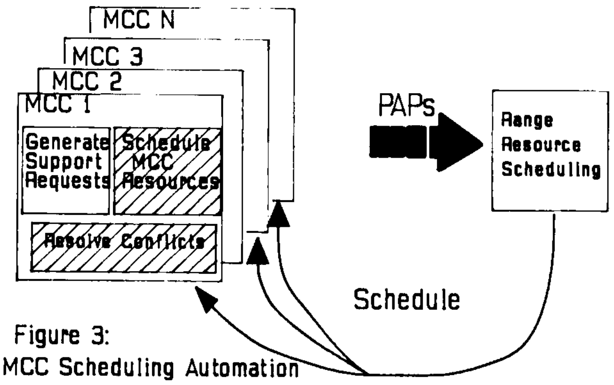


Figure 3:
MCC Scheduling Automation

project. The three projects were: MCC Scheduling Automation (MSA), Request Generation (ReGe) and ReGe II.

The MCC Scheduling Automation (MSA) research resulted in a prototype which simultaneously schedules all requests for an MCC for its resources for a specified week [Figure 3]. MCC resources considered by the algorithm include personnel and equipment such as control points, command and telemetery CSEGS, processor loading and mission unique equipment. This also includes range conflict checks. An interactive capability allows the operator to make request changes and re-generate the schedule.
The MSA project was done on the host using APL with the anticipation of interfacing with the configuration controlled databases. ReGe I took the next step in the research toward developing MCC scheduling tools for distributing the data and the processing to intelligent workstations [Figure 4].

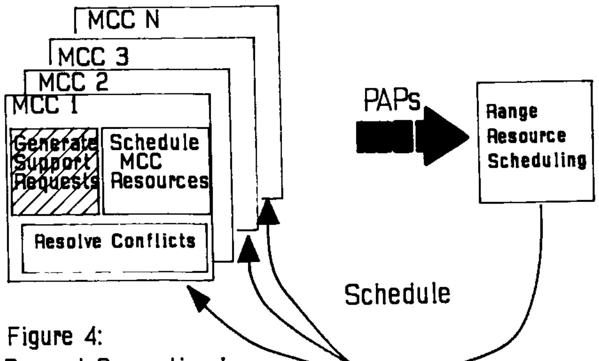


Figure 4:
Request Generation I

Request Generation (ReGe I) is the next generation MCC scheduling tool. It is a PC based prototype with a modular design which generates requests for a single vehicle per run [Figure 5]. With the lessons learned on scheduling algorithms and user interfaces behind this research, ReGe I focuses on another aspect of the problem, ´can the rules for scheduling different support or maintenance requirements be represented in a knowledge base?´ Our early prototyping in ReGe I indicates that the answer was ´yes´.

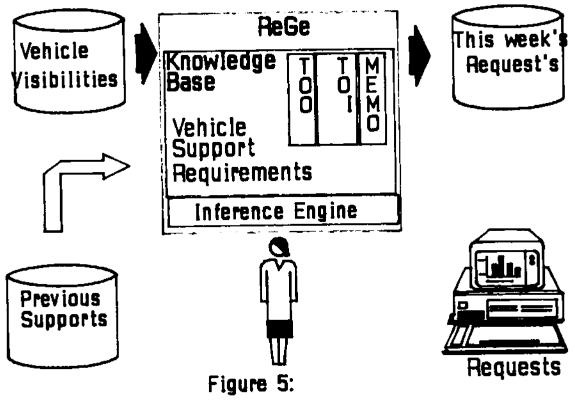


Figure 5:
ReGe Structure

ReGe II research redefines and builds on the objectives of MSA and ReGe II [Figure 6].

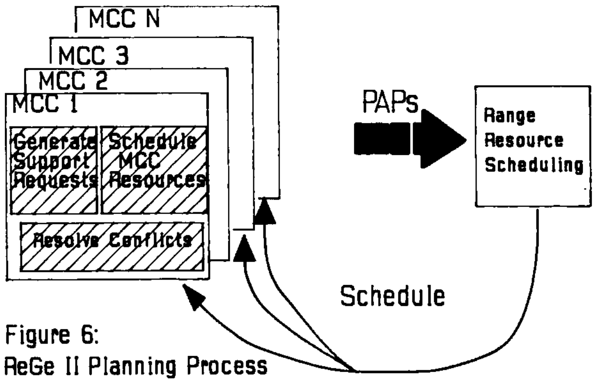


Figure 6:
ReGe II Planning Process

Objectives for ReGe II included providing a tool to generate vehicle support requests which would:

- o provide an interface to the planner analyst to
  - (a) define data to be used in the scheduling algorithm,
  - (b) initiate the algorithm and
  - (c) modify the results of the algorithm.
- o consider constraints such as vehicle events and MCC resources when applying the scheduling algorithm.
- o utilize vehicle definition and requirement definition when applying the scheduling algorithm.

The most challenging design issues of this work are the structure of the knowledge base and distribution of the interaction between the knowledge base and conventional code.

Processing schedules involving dates and numbers was found to be best performed by conventional code. Access to the data to be processed and the organization of that data was better represented at a higher level, using expert systems. The kind of data needed in the processing, thus what types of matching would be needed drove the design of the knowledge structure for vehicle definition, vehicle event definition, support requirement definition and support requests.

The greatest challenge in structuring the knowledge base for ReGe II was representing the vehicle requirements. Each requirement definition includes:

- o what event(s) drive the need for the requirement.
- o an indicator of the complexity of scheduling a requirement.
- o rules for combining requirements together.
- o which vehicles were defined as needing the requirement.

Driving events for a requirement would be one or a combination of the following: date specific, time specific, previous support(s), vehicle event(s). Again, vehicle events that a requirement might be dependent on might be acquisition or sun-vehicle-earth angles or eclipse information.

The complexity indicator of a support requirement must be reflective of the types of constraints which are on the requirement. The more constraints, the more difficult it is to schedule, the higher the complexity indicator should be. The complexity indicator is used in determining the order in which requests are generated.
Rules for combining requests for support are essential in meeting one of the primary goals of a planner analyst. That goal is to reduce the load on the satellite network resources by combining requirements and use of resources where possible into one request.

The algorithm employed by ReGe II begins by processing each requirement in priority order, for each vehicle defined as needing that requirement. A search or match is then done to find all of the events defined as triggering the need for the requirement. Once a requirement is determined to be needed for the current week, a search is then done for other support requests that this requirement can be combined with. The request is then either added to another request or generated as a new request. Lastly, resources are checked and if a conflict is identified, the request is marked as in conflict.

The user interface concepts of MSA for graphical representation of the vehicle events and the support requests and an interactivity were incorporated into ReGe II.

Beyond ReGe II, The next step is to develop a prototype of the tool for user feedback on: the HCI, the knowledge representation and the processing of the expert system and conventional code. The planner analyst needs to keep control of essential tasks and be freed from mundane, repetitive, data sorting task.

Summary

The research performed by IBM in the scheduling arena has resulted in insight and refinements to information representation, information processing and operator interaction.

IBM's research in MSA, ReGe I and II is aimed at assisting the planner analyst within the MCC by evolving the task away from manual and repetitive data sorting through automation in order to free them up for a more important emphasis: understanding the changing needs of the vehicle. In an environment where the staff is turning over frequently due to job rotation and where vehicles are increasing in number and complexity, tools which reshape the current planner analyst task are essential.

This research supports an approach which distributes data between existing configuration controlled mainframe databases to intelligent workstations in the MCCs which process data conventionally for computational operations, but represents and accesses it through the higher level representation using expert systems.

Arbabi, Mansur PHD., Garate, John A., Kocher,Donald F., "Interactive Realtime Scheduling and Control," PROCEEDINGS OF THE 1985 SIMULATION CONFERENCE, Chicago. Ill., Jul, 1985.

IBM-FSD, "Data System Modernization : System Scheduling," (AT-13), document no. C27-0020-13-01, Apr., 1980.

IBM-FSD, "Data System Modernization : Generic Range Scheduling Functional Analysis," (AT-13), contact no. F04690-81-C-003, Oct., 1982. 1980.

IBM-FSD, "Range Scheduling AAutomation," Report no. 2G59-IRAD, 1982.

N91-20677

The Ames-Lockheed Orbiter Processing Scheduling System

Monte Zweben
NASA Ames Research Center
M.S. 244-17
Moffett Field, California 94035
zweben@pluto.arc.nasa.gov

Robert Gargan
Lockheed AI Center
3251 Hanover St. O/9620 B/259
Palo Alto, California 94304
gargan@laic.lockheed.com

Abstract

This paper describes a general purpose scheduling system and its application to Space Shuttle Orbiter Processing at the Kennedy Space Center. Orbiter processing entails all the inspection, testing, repair, and maintenance necessary to prepare the shuttle for launch and takes place within the Orbiter Processing Facility (OPF) at KSC, the Vehicle Assembly Building (VAB), and on the Launch Pad. The problem is extremely combinatoric in that there are thousands of tasks, resources, and other temporal conditions that must be coordinated. We are currently building a scheduling tool that we hope will be an integral part of automating the planning and scheduling process at KSC. Our scheduling engine is domain independent and is also being applied to Space Shuttle cargo processing problems as well as wind tunnel power scheduling problems. The significant technical contributions of our scheduling system are 1) the ability to handle dynamic rescheduling while considering the time it takes to reschedule, the optimization criteria in the domain, and the amount of perturbation to the original schedule; 2) the ability to represent arbitrary state conditions that change over time and the ability to declare the requirements and effects that activities have in relation to these conditions; and 3) the explicit representation and use of search control knowledge so that domain information can drive the scheduling process. Our scheduling engine is a constraint-based system implemented in CommonLISP that runs on a variety of platforms. We have tested our system with real orbiter processing data and have found the results promising. In the near future, we plan to deploy an early prototype of the system which will be used to shadow the current scheduling process at KSC.

1 Introduction

1.1 Description of Problem

Millions of people see or follow shuttle launches each year. They are familiar with the kinds of work performed by the shuttle crew. Most, however, are unaware of the amount of work that's involved in preparing a shuttle for launch. Preparing shuttles for launch requires successful and timely completion of many operations performed by many people.

Kennedy Space Center currently uses a three-tiered approach to developing schedules for shuttle flights. At the top level is the long range schedule. This schedule represents multiple shuttle flights over several years. The middle tier is developed about 60 days prior to the beginning of the flight. At this time, the planning person develops a flow that represents all of the activities that must be performed on the specific oribiter prior to launch. The granularity of activities developed at this tier is generally one OMI per activity. An OMI (Orbiter Maintenance Instruction) essentially describes a process that must be performed. This activity generally can be broken into about 10 primitive operations that must be performed on the floor. The third tier represents these primitive operations. Due to the large quantity of operations and the liklihood of change, the third tier schedule is generated each day for the next week.

The scheduling process works as follows. Approximately 60 days before the beginning of a flight, high level planners create the middle tier schedule. They are currently using a variant planning approach. That is, they are starting from a pre-existing flow, removing work that was unique to the previous flow, adding new work specific to this flow, and then rescheduling the activities. Once finished, they perform CPM analysis to develop a schedule. This schedule has many resource constraint violations that must be resolved. The planner person then uses the target start dates and resource balancing to make the schedule to work.

Once the flow has begun, the planning and scheduling people keep a detailed 72 hour schedule. This schedule shows all activities that are being performed. Scheduling at this level is primarily done based on past shuttle flights and via daily scheduling meetings. During the meetings, representativies of the various work groups discuss their resource requirements and target completion times. The person in charge of the meeting coordinates the dynamic rescheduling of the work to be performed. Unfortunately, delays still occur. For instance, on one occasion work that was scheduled could not be performed because the necessary quality assurance people weren't available. Other times, weather could cause a delay in certain work. These kinds of delays are part of the reality of executing the schedule. In this dynamic changing environment, it is all the more imperative that the people in charge be able to see potential impact of decisions being made. KSC managers do a super job currently, given the amount of information they have. It is desirable to recognize a problem and be able to work with the system to determine a solution.

1.2 Why Use A Heuristic Approach

On the surface, it appears that a good project management tool is all that is required to manage the scheduling of activities. KSC has been doing this at the second tier level and beginning to do this at the third tier as well. Unfortunately, the project management tools can only address part of the problem. Each activity has temporal requirements, resource requirements, and configuration requirements. Existing project management tools can represent most of the temporal requirements and some of the resource requirements, however, no tool can represent the configuration requirements. Given this, the best any conventional tool can do is give you partial information.

For instance, there are activities (hazardous operations) that require the area to be cleared. When these activities occur, most other work can not occur. However, there may be no requirement saying that the hazardous operation must occur before or after other work. Conventional systems have no way of expressing this kind of temporal constraint.

Additionally, much of the work being performed on the orbiter requires that the orbiter be in a specific configuration. Again there may not be any hard temporal requirement connecting several activities, however, one activity may change the state of the orbiter and later activities may require that configuration. A simple example of this is requiring that the orbiter bay doors be closed to do certain types of tile work.

Heuristic approaches to scheduling are not new. ISIS [Fox83] and then OPIS [SFO86] focused on developing a constraint based job shop scheduling system. KSC has also had scheduling work done previously. Empress [HJK*85] and Phits [Gar87] both focused on different aspects of planning and scheduling of cargo processing.

1.3 Our Approach

We have viewed KSC scheduling as a Constraint Satisfaction Problem (CSP). In our system, we represent most information using variables. Each variable can take on a range of values. Constraints are used to filter the values of the given variables. For example, Figure 1.3 shows two activities and some of the variables associated with them. We use constraints maintain the relationship between the start time, the end time, and the duration of an activity. Additionally, using constraints we can express a requirement that one activity must start before the other one can start (or similarly with the end time) or that one must precede the other. During the scheduling process, the list of possible values for a given variable is filtered based on the various constraints. The scheduling system searches the space of possible schedules for a time when all variables can be fixed and all constraints satisfied.

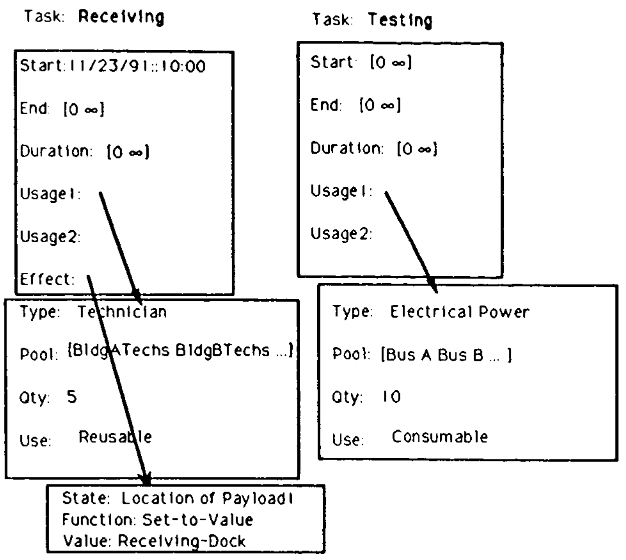


Figure 1: Task Representation and the Use of Constraints

The remainder of this paper describes the process in more detail. We first introduce the knowledge representation we have utilized. This description will be of the various types of constraints we are using. Next we describe a rule system we have recently added that allows the user to encapsulate the search control knowledge explicitly rather than implicitly as is usually done. Finally, we describe the process of rescheduling. Rescheduling is especially critical to KSC since activity status is in a constant state of flux.

2 Knowledge Representation

Scheduling knowledge is being represented via constraints. Constraints are applied to the various variables that are a part of the objects of the system. For instance, our major object in the system is the activity. Activities contain many status slots. They also contain slots represented as variables such as the start time, end time, and duration. Constraints can be tied between multiple variables to maintain some consistency between various objects. For instance, a constraint would be used to state that a resource that is needed for a specific activity must be available during the time of allocation. The remainder of this section gives a brief overview of the types of constraints we represent.

2.1 Resource Requirements

Constraints can be used to require specific resources for an activity. Resource classes can be represented hierarchically. Each resource class can have one or more resource pools associated with it. Each resource pool can have a capacity of 1 or more. A resource pool of one is used to represent a specific individual or piece of equipment. Alternatively, if uniqueness is not of concern the pools could contain many values. Using this type of resource, would result in allocating one from the pool to the activity, however, it wouldn't matter which one.

2.2 State Requirements

An important distinction between our system and others is our representation of state information and the use of constraints to maintain the proper state. For instance, most of the work on the shuttle requires that the orbiter be in some particular state (for instance, the orbiter doors being closed). As was mentioned before, current systems cannot represent this information. In our system, the activity representation has been extended to support task requirements and task affects in addition to the representation of states.

A state is an object in our system that can take on multiple values. We will eventually support finite state machines, although we currently support known state changes. For instance, the orbiter contains two bay doors. Each door can take values of opened, closed, or half-open. Other state information is represented similarly. Activities now can require that before the activity can start the object (in this case the shuttles bay door) must be open. Conversly, an activity can specify that as a result of executing the given activity the following object state will be changed. In the above example, an activity might specify that the pay doors are moved from the open to closed position.

The task requirements and affects described above are encoded as constraints on the given activity. These constraints must be maintained by the system in the same manner as the other constraints. By representing this type of information explicitly, the scheduler can take advantage of this type of knowledge when sequencing operations. Scheduling systems that have been developed previously would have represented this information implicitly as precedence constraints. This would reduce the overall flexibility of the system as well as make it more difficult to reschedule activities when problems arise. By using the state information to constrain the activities, it allows a more flexable schedule.

2.3 Temporal Relationships

Even though some information should be represented via state constraints, other information still should be represented via temporal constraints. Some of the most often used constraints are the precedence constraints. One type of precedence relation states that activity-1 must be completed before activity-2 can start. All standard off the shelf project management tools allow this form of representation. Additionally, some tools will also allow the user to specify that one activity must begin before the other one can begin (or end before the second one ends). The standard tools, however, do not generally allow the user to place delays on all the precedence relationships. In addition, they don't allow the user to express that two activities can occur in any order, however, they can not happen at the same time. This form of mutual exclusion is necessary for constraining hazardous operations on the space shuttle. This type of work must be done without other work being done in the area.

2.4 Calendars

Finally, it is necessary to represent when the work can actually be performed. We represent this type of information in a calendar. A calendar specifies when work can be performed. It takes into consideration holidays and daylight savings time. Each activity specifies a calendar that should be used to determine when it should be scheduled. This is consistent with the current way in which work at KSC is scheduled. There are a variety of calendars ranging from one 8-hour shift 5 days a week to a 7 day 24 hour calendar.

2.5 Representing Control Knowledge

One of our goals in developing this system was not to tie the scheduling process too closely to the KSC domain. Different applications should be scheduled in different manners. There is always domain knowledge that can be utilized to more effectively schedule operations. There is a basic conflict here between adding in this domain specific control knowledge and still maintaining generality.

Our desire would be to use a formal language to specify the control knowledge necessary to guide the scheduler through the search process. This would allow the applications developer to add specific control knowledge to the system to customize it for their application. Currently, most scheduling systems use LISP code to encode the search strategy. The problem with this is that it implicitly requires the search process to be the same for all applications of the scheduling system. Even if a single application area is all that was intended, this approach also fails because it restricts the end-user from being able to customize the search process at some later date should the need arise.

As a result of these issues, we are integrating a rule language into our scheduler. The user will encode all search control knowledge into the rule system. The system presently contains general knowledge encoded in rules. For instance, one rule that plays a role in determining the task to schedule states that if there is a same start or same end time relation between two tasks and the second task is not scheduled, then strongly prefer scheduling this task next.

As we continue to develop this portion of the system, we will be adding more KSC specific rules into the system. For example, there are certain types of work that are prone to identifying unforseen problems, so it is useful to do this work as early as possible so the problems can be identified. This type of domain specific information will eventually be represented as a rule to the system.

3 Dynamic Rescheduling

One of the most critical needs of KSC is the ability to reschedule the activities because work schedules are constantly changing. There's a variety of reasons ranging from unanticipated training schedules to bad weather. Whenever a problem arises, it is necessary for the planning people at KSC to adjust schedules to handle the problems proposing the appropriate work around. While this is by far their worst problem, it is also one which is extremely difficult to meet using the existing tools. Because of the lack of support for the various kinds of constraints described above, planning personnel have used precedence relationships to force the schedule to take place in a certain order. Doing this, however, reduces future flexibility in the schedule by arbitrarily using precedence relationships where they are not really necessary Additionally, the planning personnel force the start time for many activities in the schedule to be fixed rather than relying on the constraints to determine new start times. This often results in the work not being performed when scheduled because the orbiter is not in the appropriate configuration or the necessary resources are not available. The resolution of these problems are often the subject of the daily scheduling meetings involving many KSC and Lockheed personnel.

Our goal with our scheduling tool is to provide the rescheduling capabilities to alleviate the above problem. In order to be successful, we must provide a tool that efficiently determines the new schedule. Our approach has been to investigate the use of iterative improvement scheduling algorithms. This approach differs from traditional "AI" scheduling approaches in that they incrementally repair complete solutions to the scheduling problem rather than systematically extending the partial solution to the problem. Our approach has led us to develop a framework [Zwe90] that converges on a solution by making local repairs to the violated constraints of some approximately correct schedule. This approach has two advantages over the other approaches. First, our approach is significantly faster than conventional heuristic based scheduling techniques. Second, because of the nature of our algorithm, a solution can be returned at any point in the algorithm, with the solution improving the longer the algorithm is given to execute.

Our algorithm is implemented in two phases. The first phase is the systematic repair of all temporal constraints. The result is a schedule that is consistent with respect to temporal constraints, but is likely to contain resource and state varable constraint violations. This schedule is the input to the second phase – constraint-based simulated annealing. During this phase, the scheduler incrementally repairs violated resource and state-variable constraints. The remainder of this section describes the two phases in more detail.

3.1 Temporal Shift

The temporal shift, which is the first phase of our rescheduling algorithm, takes a desired change in start and end times for a given activity and creates a schedule without any temporal constraint violations. We originally achieved a consistent schedule by systematically shifting all activities with temporal constraint violations in a fashion similar to those used by OPIS [OST88]. We later discovered that this approach by itself would not fill our needs because of constraints tying the end time of one activity to the end time of another (or start time to start time constraints). These constraints, in conjunction with the more conventional constraints (end time to start time) could lead the system back to the original task that was moved, so the approach of taking the earliest unscheduled task would no longer apply.

We decided to use Waltz's algorithm [Dav87] to address this anomaly. The algorithm is based on changing the intervals for each activity when shifted. Each change causes the interval to be filtered so that each interval continues to represent the range of times when an activity can begin (or end). The algorithm begins by rescheduling the changed task. It then collects the activities that have temporal constraint violations. Those activities' times are then filtered by a similar amount. This algorithm has the advantage that it quickly determines plausable schedules with minimum amounts of change and works for the general class of constraints used by KSC.

The algorithm is not guaranteed to be successful. If an activity has been marked as permanent and an attempt is made to move it, then the algorithm will return unsuccessfully. This could be useful for addressing milestones as well as activities dependent on some natural event (i.e. sunrise).

3.2 Constraint Based Simulated Annealing

The second phase is based on simulated annealing [KGV83]. It begins with the scheduling assignment resulting from phase one of rescheduling and then evaluates a "cost" of the assignment. The cost function for our experiments is the number of constraints violated for the given assignment. Then, by repairing constraints, it suggests a new solution and evaluates its cost. If the new cost is an improvement, it adopts the new assignment and continues. If the new solution is worse, the algorithm adopts it with some probability. This last step allows the algorithm to escape local minima. We have customized this general approach to constraint satisfaction problems which is described in more detail elsewhere [Zwe90]. The basic algorithm is as follows (where T is a set of tasks with assignments made in phase one):

```
Solve(T){
   Old = Cost(T);
   Repeat until Old <= *THRESHOLD* {
      Next = Find_New_Solution(T);
      New = Cost(Next);
      If New < Old Then { Old = New; T = Next;}
                 Else { With probability P do
                          { Old = New;
                            T = Next;} };
      SaveBestSolutionIfNecessary;
   }
}
```

3.2.1 Systematic Repairs: Finding a New Schedule

In our previous work, we concentrated on simple local repairs in order to investigate the utility of the simulated annealing search framework. Here, we focus on fast rescheduling, with a heuristic bias against schedules with excessive work-in-process (WIP) time and against schedules that require radical perturbations to the original schedule. This bias is enforced by the repair strategies themselves. First, only those tasks involved in constraint violations are modified, and second, when tasks are moved they are not moved drastically.

Our repair strategy also exploits the knowledge that any task move is likely to violate temporal constraints. Thus, after any constraint repair causes a task to move, temporal constraint violations are resolved first by executing the temporal shift algorithm given above. Because these repairs explicitly exploit the knowledge of how repairs interact, they are no longer local.

The following are two of the repair strategies employed by the rescheduler:

```
capacity(?start ?end ?resource):
1. Deallocate this current resource.
2. Try to find a pool that is available
       from ?start to ?end.
3. If one exists, change ?resource to
       be that pool and reallocate.
4. Otherwise task = the task associated
       with this constraint;
     new-start = ?start + random(1 .. 10)
                 * c * d;
     new-end = new-start + duration(task);
     TemporalShift(?task, new-start, new-end);
```

The constant c is a small,fixed time unit (a day in the payload processing domain) and d is a direction (1 or -1) that is set by the change that the user makes. The strategy attempts to substitute a new resource pool, but if that is impossible, it moves the requesting task back or forward in time. After the task is moved, the temporal shift algorithm of phase one is executed – this systematically propagates the change caused by the repair to all temporal dependents.

```
temporal-equals(?t1 ?t2 ?a ?v):

First strategy:

1. supporter = the first task after ?t1
               that sets ?a = ?v;
2. task = the task associated with this
          constraint;
3. new-end = start(task) - c;
4. new-start = new-end - duration(supporter);
5. TemporalShift(supporter, new-start, new-end);

If unsuccessful:

1. task = the task associated with
          this constraint;
2. new-start = the first time of a state
               transition, t,
     (away from ?t1 in the direction of d)
     where ?a is set to ?v;
3. new-end = new-start + duration(task);
4. TemporalShift(task, new-start, new-end);
```

This repair is analogous to the modal truth criterion of non-linear planning [Cha87] but without the flexibility of adding actions. The preferred repair is to move a task that sets the state-variable appropriately, to a time interval before the task that has the requirement (i.e., to use Chapman's terminology, moving a white knight). If this is impossible, the task with the requirement is moved to a point in time when the state variable is set appropriately. This will move the task directly after the closest white knight.

In either case, to perform a move, the temporal shift of phase one is employed which results in a temporally consistent schedule.

4 Development Status

The project to apply the scheduler to the KSC shuttle processing problem has been underway for about a year. Since February, we have been working with actual data from a completed shuttle flight. While this data did not provide us with the data to utilize our state variable representation, it did provide us the ability to test our algorithms on realistic amounts of data.

Our plan is to shadow the STS-37 flight later this summer. The initial purpose of this first test is to collect the necessary scheduling information to put into the system. Currently, no information exists in computer form stating various configuration requirements of the orbiter for the various activities. Additionally, the resource information that is presently stored must be compared with the floor supervisors for accuracy as well as adding new resources that are not presently being stored. During the testing period, we will add in the changes to the work as they occur providing new schedules in a timely manner. As the quality of the information being stored in the knowledge base increases, our system will produce better schedules. The schedules we produce will then be compared to existing work schedule providing us some insight into new information to add to our system. Our hope is that even at this early phase of testing, we will be able to provide the KSC personnel some insight into alternative schedules that might not have been considered in the past.

5 Conclusion

In this paper, we described a research scheduling tool that is being applied to scheduling ground processing activities for the space shuttle. Research in this area has been on-going for several years and is at a state where an application of this magnitude can be attempted. We provided a brief overview of the scheduling system providing examples of the use of the various pieces to the KSC application. Experimentation with the repair strategies will continue as we use the rescheduling component of the system with the real data. It is generally felt, that there is a tremendous potential for savings to the shuttle program if this effort and the other phases of the scheduling process (not described here) at KSC are automated.

References

[Cha87] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(4), 1987.

[Dav87] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(4):281–331, 1987.

[Fox83] Mark S. Fox. *Constraint-Directed Search: A Case Study of Job Shop Scheduling.* PhD thesis, Carnegie-Mellon University, 1983.

[Gar87] R.A. Gargan Jr. Mission planning and simulation via intelligent agents. In *Proceedings of Space Station Automation III*, November 1987.

[HJK*85] G. B. Hankins, J. W. Jordan, J. L. Katz, A. M. Mulviehill, J. N. Dumoulin, and J. Ragusa. Empress: expert mission planning and replanning scheduling system. In *Proceedings of Expert Systems in Government Symposium*, 1985.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 1983.

[OST88] P. Ow, S. Smith, and A. Thiriez. Reactive plan revision. In *Proceedings of AAAI-88*, 1988.

[SFO86] Steven F. Smith, Mark S. Fox, and Peng Si Ow. Construction and maintaining detailed production plans: investigations into the development of knowledge-based factory scheduling systems. *AI Magazine*, 7(4), Fall 1986.

[Zwe90] Monte Zweben. A framework for iterative improvement search algorithms for constraint satisfaction problems. In *AAAI-90 Workshop on Constraint-Directed Reasoning*, 1990.

N91-20678

AN INTELLIGENT VALUE-DRIVEN SCHEDULING SYSTEM FOR SPACE STATION FREEDOM WITH SPECIAL EMPHASIS ON THE ELECTRIC POWER SYSTEM

Presented by: Joseph C. Krupp

Decision-Science Applications, Inc.
1110 North Glebe Road, Suite 400
Arlington, Virginia 22201

ABSTRACT:

This paper discusses the Electric Power Control System (EPCS) created by Decision-Science Applications, Inc. (DSA) for Lewis Research Center (LeRC). This system in its current form makes decisions on what to schedule and when to schedule it, including making choices among various options or ways of performing a task. The system is goal directed and seeks to shape resource usage in an optimal manner using a value-driven approach. The paper discusses the considerations governing what makes a "good" schedule; how to design a value function to find the best schedule; and how to design the algorithm which finds the schedule that maximizes this value function. Results are shown which demonstrate the usefulness of the techniques employed. The value-driven approach also allows for the system to be easily extended to an emergency response system, making decisions as to where to best cut power when warranted.

1.0 DEFINITIONS

1.1 Activities

The EPCS schedules **activities**, **tasks**, **options**, and **subtasks**. These terms have very specific meanings with regard to the scheduler and are defined as follows:

Activity. A group of tasks directed toward a single goal, e.g., "Core Activities" or "Biology Experiments". Each activity has a value or priority.

Task. A well defined part of an activity, e.g., "Metallurgy Experiment 1". Each task is independent of all others, i.e., there is no specific order in which the various tasks must be completed. The tasks may even be done simultaneously. However, each task can have a time window, meaning that the task must be performed sometime within a particular time interval. The time window is not due to dependence of the tasks, but rather due to the nature of the task (e.g., it must be done during an eclipse period). [Actually, the time window is associated with the "option"--see below.] A task may be either a single time task, or may be a task which should be repeated periodically. The period of a task may be defined in terms of hours, orbits, or days. Each task has a value associated with it expressed as a percentage of the value of the activity of which it is a part. The highest priority task always receives 100% of the activity value. Less desirable tasks may receive a lower percentage of the activity value.

Option. A way of performing a task. Different options may have different numbers of subtasks, and will usually have different resource profiles. The options may also have different time windows. For example, a surveillance activity may have several windows of opportunity during which the surveillance can occur. Only one option for a given task is scheduled. Each option has a value associated with it expressed as a percentage of the value of the task. The best option always receives 100% of the task value. Less desirable options receive a lower percentage of the task value. For real time control (i.e., emergency response) it is important to know not just the value of a completed option, but how this value accrues. The actual value accrued as the option is performed depends on how much of the option is completed. Each option has a defined function describing the amount of value obtained (as a percentage of total value) as a function of percent completion of the option. The fraction of completion is based on the currently completed subtasks associated with the option.

Subtask. A well defined part of an option. The subtasks must be performed in a particular order. However, the time between subtasks may be variable. There may be a wait period before which the next subtask cannot be started, and a relative time window in which the next subtask must be completed. Each subtask is classed as non-restartable, restartable, or interruptible. If a non-restartable subtask is aborted, the task (option) of which it is a part cannot be completed and all subsequent value associated with that task is lost. If a restartable subtask is aborted, then it may be restarted from the beginning as if it had never been scheduled in the first place (assuming it can do so within its time window). If an interruptible subtask is aborted, it may be restarted at the exact point it was aborted without loss. The percentage completion for the activity of which it is a part is based on the percent completion for an interruptible subtask. For other classes, the subtask must be completed before the percent completion is increased.

Note that the values associated with activities, tasks, and options are best thought of as being set independently as if by a Vice President (activities), a department head (tasks), and a project manager (options).

1.2 Resources

The EPCS currently recognizes three types of resources:

Assignables. Assignable resources are those which are used in discrete units and which can be reused by another subtask after being released by the subtask currently using them. Examples include crew and workstations.

Consumables. Consumable resources are those which are used in arbitrary amounts and which are destroyed (or created) on use. Consumables may be produced by a subtask instead of consumed. For example, electrolysis consumes water and electricity, but produces oxygen and hydrogen.

Specifics. Specific resources are those which are a particular kind of generic resource. For example, the crew may contain specialists. One activity may need a metallurgist while another may need any crew member. The metallurgist is a specific type of crew, which is a generic assignable. Using the metallurgist reduces both the number of metallurgists available and reduces the number of crew available.

Other types of resources may be defined but are not currently incorporated in the EPCS. For example, one type of resource is a "state". Some subtasks may generate a vibration state which prohibit certain other activities from functioning. This resource type is currently being considered for addition into the EPCS.

Note that electric power is both an assignable and a consumable. From the concept of *power*, it is an assignable--only so much power can be drawn at any time. From the concept of *energy*, it is a consumable, since the batteries can hold only so much energy.

2.0 WHAT CONSTITUTES A GOOD SCHEDULE

The *primary* purpose of the EPCS is to schedule activities in such a way that the productivity of Space Station Freedom (SSF) is enhanced. In its simplest form, this translates to solving a knapsack problem. That is, if one schedule allows a certain set of tasks to be performed and a second schedule, by moving the subtasks around, makes room for one more task to be performed, then the second is better. But this is a simplistic view of things, and in reality tradeoffs must be made. The first and most obvious tradeoff is that not all activities are as important as others. Thus, the notion of values comes into play. If values are assigned to the activities, tasks, and options, then that schedule which allows a set of activities with a total higher value than that of another set of activities is clearly better. So far, so good. But there are other tradeoffs.

All options have a time window associated with them (which may in fact be the entire planning time) during which they may be performed. Usually there is some preference as to when in this time window it would be better to schedule the option. In most cases, this is at the beginning of the window--due to the possibility of unforeseen problems, it is better to be early than late. Two schedules may schedule the exact same set of options. The first, however, may have all options being performed early in their time window while the second may have some options being performed late in their time window. One would judge the first schedule as better than the second. Thus, some value must be lost the longer an option is delayed.

In a similar vein, many tasks are periodic, i.e., they need to be scheduled on a regular basis. If a task is to be scheduled on a daily basis, one would prefer a schedule in which the task is performed at roughly the same time every day to one in which the task is performed late in the day one time and early in the day the next.

The scheduler must consider two aspects of the power system: battery charge and power flow. Both of these aspects are important. Consider two schedules which are identical with regard to the tradeoffs discussed above. If in the first the battery is drawn to a dangerous level of discharge while in the second the battery is always well charged, then the second is clearly better. Similarly, if the first has periods of very high power consumption followed by periods of very low power consumption, while the second maintains a relatively constant power flow, then the second is better. This is because I^2R losses for the first schedule will be higher.

Assigning values to the activities is not a problem affecting the design of the scheduler. The user may assign values to the various activities, tasks, and options in any way he wishes. The other tradeoffs do pose a problem, however, because a value function must be devised such that various tradeoffs are properly balanced. For example, if the battery charge and power flow considerations dominate, the best schedule may be to do nothing. Then the battery could stay happily charged and the power distribution system could stay cool. But this hardly enhances productivity! Similarly, the purpose of specifying a time window for an activity is that it is acceptable to delay the start of the activity, so the value lost by delay should not be great enough to prevent moving the activity in order to allow an additional activity to be scheduled.

3.0 COSTS VERSUS VALUES

The EPCS is a value-driven scheduler and emergency response system. By value-driven is meant that decisions are made which maximize total value returned, i.e., profit. In the previous sections we discussed the *intrinsic* values of an activity, task, and option, and discussed various tradeoffs that need to be considered in choosing one schedule over another. In this section we will describe the notion of *costs*, and the role they play in quantifying or codifying these tradeoffs. The value function to be maximized is represented as value minus costs, and it is the functional form of the costs which codify the tradeoffs to be made. Costs are of two types: 1) resource costs (the marginal cost of an additional unit of resource), and 2) opportunity costs (which relate to the time placement of the subtasks).

3.1 Resource Costs

3.1.1 Nominal Costs

All subtasks use resources. We would like to define the concept of cost for a resource in order to quantify the profit gained by performing a particular option. The cost concept is intuitive and easy to understand--options using more costly resources may be less preferred to options using less costly resources depending on the relative intrinsic value of the two options. But how do we define the cost of a resource? To a first approximation, each resource can be thought of as having a *nominal cost* inversely proportional to the amount normally used during a planning period. That is, if the Space Station is sized to use X units of nitrogen and Y man-days then the nominal cost of nitrogen is $1/X$ and the nominal cost of a man-day is $1/Y$. In this way, the total nominal cost of every resource during a normal planning period is 1.0. This normalization reflects the fact that the Space Station was sized intelligently and provides an intuitive quantification of the marginal cost of a unit of resource *in the Space Station environment*.

It should be noted that in the real Space Station environment these nominal costs would be provided to the EPCS by the individual control systems. This will allow the individual systems to make allowances for special situations. For example, a consumable resource which was being used at a much slower rate than usual could have its nominal cost lowered, while one which was being used more quickly could have its nominal cost raised. In the prototype system, the nominal costs are set by considering the total amount of resource needed for all options as a surrogate for the amount normally used in a planning cycle.

3.1.2 Cost/Benefit Ratios

The use of nominal costs also allows for an interpretation of the intrinsic values of an option as a benefit/cost, or profit/cost, ratio. This ratio is more what the user has in mind when he assigns values to the various activities, tasks, and options. If one option has an intrinsic value twice that of another, the user expects that option to be scheduled over the other if possible. If the nominal costs (i.e., the sum of all resources used times their nominal costs) of the two options differ substantially, this may not be the case. The definition of the user-supplied values as profit/cost ratios lets the user assign these values without needing to know the nominal cost of performing the option. Thus, the actual value of an option used in the EPCS is the product of (one plus) the assigned value times the nominal cost of the option.

3.1.3 Cost Curves

Any resource which has a supply much larger than required to perform all subtasks is a non-player with regards to any scheduling decision which is made. In the real world, such a resource would be free ("You can't even give that stuff away."). If all resources were like this, one would maximize total value by scheduling the most preferred options of every task. This is not usually the case, however. Any resource which is in short supply will have a cost associated with it which reflects the balance of supply and demand. That is, there is a cost curve associated with each resource. Resources which use more than the nominal amount for the planning period will have a higher cost than resources which use less than the nominal amount for the planning period. The cost curves are supplied by the resource control systems aboard the Space Station. We will provide the cost curve for the electric power system in detail. In the prototype, an exponential is used as a surrogate cost function for the other resources:

$$\lambda = \lambda_0 \exp[\alpha (U-R)/R] \qquad (1)$$

where	α	=	an adjustable parameter for each resource,
	λ_0	=	the nominal cost of the resource
	U	=	the resource usage
and	R	=	the amount of resource available (or nominal amount to be used during the planning cycle)

Note that for assignables, resource usage is in terms of units used at any given time, e.g., crew used minus total crew, while for consumables the resource usage is in terms of the largest deficit at any time in the future. That is, if a consumable is overused by tomorrow, I should conserve it today.

An option using a given resource will lose value equal to the cost of that resource. In fact, if the cost is so high that the total profit, value minus cost, is negative, it is preferable to drop the option from the schedule. Resource costs are therefore a function of resource use. At low use the costs are low, while at high use the costs are high. There are two aspects of these resource costs which need to be determined: nominal costs and cost curves.

Note that the cost curves do not usually go to infinity if the resource is overutilized. This is because the scheduling algorithm is an iterative one. Making the cost curves go to infinity at the constraint point, i.e., using a "brick wall" approach, will ensure feasibility but will not promote optimality. It is necessary to allow infeasibility so that a particular subtask which can move out of the way will do so. This will be made clear in a later section when we discuss the algorithm employed.

3.2 The Electric Power Cost Functions

3.2.1 The Battery Charge Penalty

We desire a battery charge penalty, or cost, designed to keep the battery reasonably well charged, yet still allow the battery to go below nominal minimum charge if absolutely necessary. This function should be zero at the charge level where trickle charging needs to begin, be 1.0 (times the nominal cost) at the nominal minimum discharge level, and rise sharply below this level. Such a function is simply:

$$\lambda_B = \lambda_0^{elec} [(C_{TR} - C) / (C - C_{min})]^{\beta} \qquad (2)$$

where C_{TR}	=	trickle rate level (.95),
λ_0^{elec}	=	the nominal cost of electricity (per kw-H)
C_{min}	=	2 C_{nom} - C_{TR} (.35 for given C_{nom} and C_{TR})
	=	absolute minimum that will not be violated under any circumstances.
and C_{nom}	=	nominal minimum discharge level (.65).

This function is plotted in Figure 1 for $\beta = 3$.

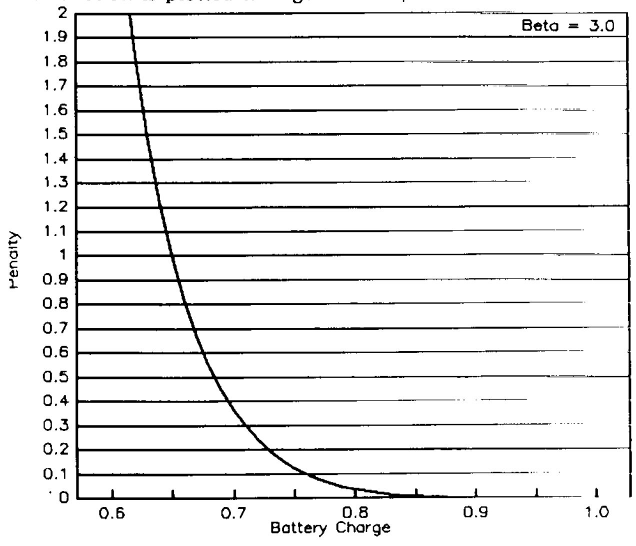


Figure 1: Battery Penalty Functions

Note that the total penalty is integrated over the entire planning time. Thus, a schedule which went below the nominal minimum discharge for a very brief period, but otherwise stayed well above it, would score better than one that stayed above the nominal minimum discharge level, but just barely above it, for a long period of time. This makes sense from an emergency response point of view.

3.2.2 The Power Flow Penalty

Power enters the scheduling considerations in two ways. For high power levels, the primary consideration is that the power not exceed acceptable safe levels. In fact, the power flow penalty should approach infinity if the power required is greater than the maximum power available (even if the battery has energy available, it is limited in how quickly it can deliver this energy, i.e., the power it can deliver is finite). For lower power loads, the primary consideration is to smooth out the power load across time. In general, this second consideration should be small enough that it does not cause a task to be removed from a schedule, but does cause it to be adjusted slightly to smooth the load.

The second consideration can be achieved through a modification of the electric power cost, λ_B, described above. Since the reason for wanting to balance the loads is to reduce I^2R losses, λ_B can be multiplied by a factor proportional to the square of the power as follows:

$$\lambda_B' = \lambda_B [1 + (\beta P)^2] \qquad (3)$$

where $\beta = [(1 - \epsilon) / \epsilon]^{0.5} / P_0$,

ϵ = Efficiency of the PMAD system at the nominal power level (0.93)

and P_0 = the nominal power level

The first consideration is no different from any other type of resource. Here power is considered as an assignable resource--only so much power can be used at any given instant. We therefore use an equation similar to Eq. (1) above:

$$\lambda_P = \lambda_0^{elec} \exp[\alpha (P - .9 P_{PV}) / P_{PV}] \qquad (4)$$

where P_{PV} = Power from the Photovoltaic Array.

3.3 Opportunity Costs

The schedule may be shaped by other considerations than how the resources are used. For example, there may be options for which there is a preference as to where in the time window the option is scheduled. Similarly, for tasks which need to be repeated on a periodic basis it is preferred that each subtask be performed more or less at the same time within each period. Although the prototype EPCS does not yet take into account any preference within the time window, it does take into account the preference for periodic tasks being performed at similar times within the period. It does so by subtracting from the profit of an option a fraction of each subtask's value[1], where the fraction is defined as:

$$1.0 - \exp[-0.5 ((t-t_0)/\sigma)^2], \qquad (5)$$

where t = time of proposed scheduling of the subtask

t_0 = desired time of scheduling
$= [(t_{-1} + T_R) + (t_{+1} - T_R)] / 2$

$\sigma = 2 T_R$

and T_R = Repeat time

i.e., a Gaussian centered around the desired time with a standard deviation of twice the repeat time.

This function is small enough that subtasks are free to move if necessary, yet large enough that, all things being equal, the subtasks will tend to be performed at regular times.

4.0 THE SCHEDULING ALGORITHM

The scheduling algorithm consists of three interacting processes:

1. The Basic Algorithm
2. Feasibility Adjustment
3. Optimality

4.1 The Basic Algorithm

The basic algorithm consists of scheduling each task in turn. During this phase, one option for each of the tasks is always scheduled. When scheduling a particular task, the scheduled option from the previous iteration is "picked up", i.e., is removed from the schedule. The resource usage is calculated with all other tasks implemented. This determines the resource cost for any subtask of any option for the task to be scheduled. Each option for the task is considered in turn and an optimal placement for all subtasks for that option is determined. The option with the largest profit (even if the profit is negative at this point) is scheduled.

Determining the optimal placement of the subtasks for an option is done with a dynamic programming algorithm. The cost for each subtask is determined for each of N delay times. By working backwards from the last subtask to the first, the optimal delays (within the time resolution of the N delays) can be determined for the entire option.

4.2 Feasibility Adjustment

When the basic algorithm has converged, i.e., the options picked and the scheduled times for all subtasks is not changing from iteration to iteration, the schedule may not be feasible. There are two reasons for this. The primary reason is the granularity of time periods. Resource usage is not stored on a minute-by-minute basis for the entire planning time. Rather, a set of time periods, or "bins", are defined and resource usage is added to these bins. At the

[1]The value of a subtask is the value of the option times the ratio of the duration of the subtask to the sum of the duration for all of the subtasks. That is, for the purpose of the <u>scheduling</u> function of the EPCS we are assuming linear value accrual.

start of the program these time periods are defined as the daylight and eclipse times of the Space Station. Usage of assignables is by units times time, i.e., man-days, kW-days, workstation-days, etc. Therefore, while a time period may contain enough crew-days to satisfy the resource usage required for the schedule, there may be a conflict in that for a short period of time, more crew than are available are needed. To fix this problem, a rule based "tweak" algorithm is applied to make small adjustments to the schedule to get feasibility if possible, and new time periods are introduced at the problem spots to keep the problem from reoccurring.

4.3 Optimality

A secondary reason for non-feasibility, of course, is that more is being scheduled than can fit within available resource and time constraints. Thus, if the feasibility adjustment does not result in a feasible schedule, the system checks for negative profit. All tasks with a negative profit are sorted from most negative to least negative. The worst task is then removed from the schedule and the resource usage adjusted. If the next task now has a positive profit, it is skipped; otherwise it too is removed. This process continues until all tasks have a positive profit. Those tasks removed will stay removed (unless a second option could be scheduled). If the schedule is still not feasible, a call is made to the resource suppliers to adjust the cost curves[2] and the model resumes with the basic algorithm.

If the schedule is feasible, an endgame phase is entered whereby the schedule is adjusted according to the basic algorithm, but allowed to move only a few minutes either way from the current schedule. This is to make fine adjustments due to the opportunity costs associated with the timing.

5.0 RESULTS

Figure 2 shows the electric power profile for a particular two day schedule. The gray areas are eclipse periods. This schedule consisted of 10 activities containing 37 tasks which had 57 options consisting of 370 subtasks. The important thing to notice is that the power used, in a gross sense, is fairly uniform, and where there are peaks, they tend to fall during periods of daylight. Figure 3 shows the battery charge for this same schedule. In Figures 4 and 5, we have plotted graphs for the same mission which were derived by removing the cost of electric power. Note that we still did not let the battery become too discharged, but the power levels are grossly non-optimal. This demonstrates the technique employed, while providing feasible schedules at a minimum, also provides an efficient means for shaping resource usage.

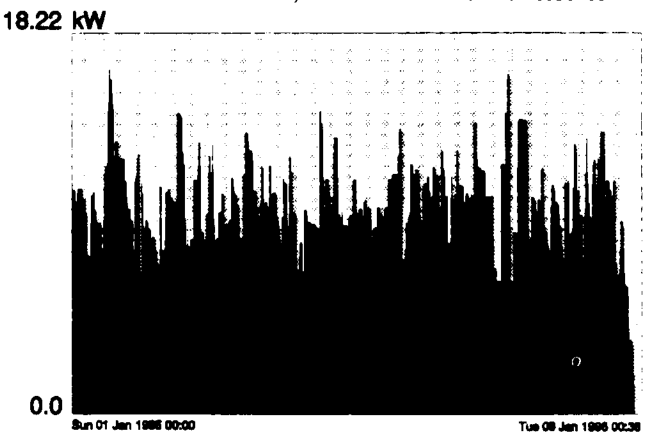


Figure 2. Electric Power Profile of Generated Schedule

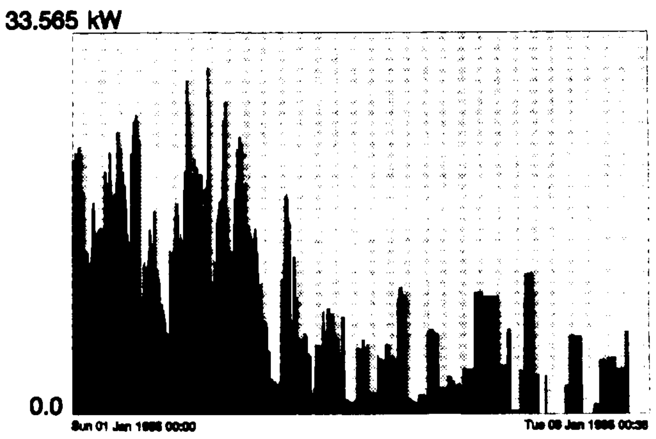


Figure 4. Electric Power Profile-No Resource Shaping

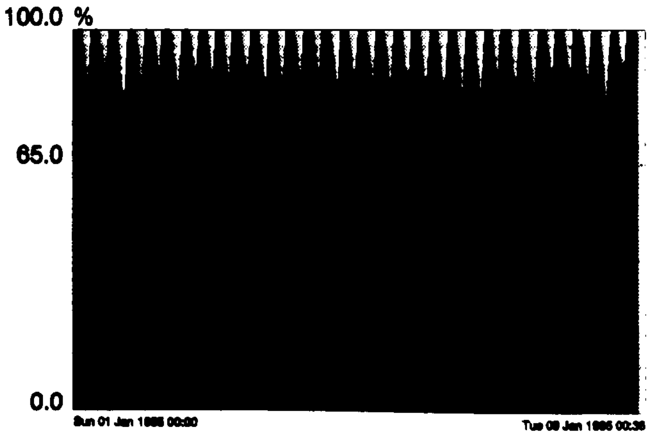


Figure 3. Battery Charge Profile of Generated Schedule

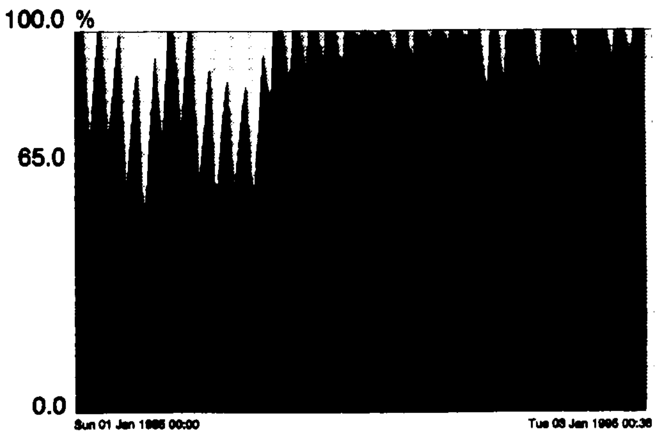


Figure 5. Battery Charge Profile-No Resource Shaping

[2] In the prototype, the value of α in Equation 1 is increased.

SPACECRAFT AUTONOMY PANEL SESSION

(No papers presented at this session)

PRELAUNCH PROCESSING

PRECEDING PAGE BLANK NOT FILMED

N91-20679

EXPERT SYSTEM DECISION SUPPORT FOR LOW-COST LAUNCH VEHICLE OPERATIONS

Dr. G.P. Szatkowski and Barry E. Levin

GENERAL DYNAMICS
Space Systems Division

5001 Kearny Villa Road
San Diego, CA 92138
(619) 496-7093

Sponsored by: USAF/Air Force Space Division

ABSTRACT

This describes progress in assessing the feasibility, benefits, and risks associated with AI Expert Systems applied to low cost expendable launch vehicle systems. This work was funded under the joint USAF/NASA Advanced Launch System (ALS) Program as applied research. Part One identified potential application areas in vehicle operations and on-board functions, assessed measures of cost benefit, and finally identified key technologies to aid in the implementation of decision support systems in this environment. Part Two of the program began the development of prototypes to demonstrate real-time vehicle checkout with controller and diagnostic/ analysis intelligent systems, and to gather true measures of cost savings vs. conventional software, verification and validation (V&V) requirements, and maintainability improvement.

The Expert System advanced development projects (ADP-2301 & 2302) main objective was to provide robust intelligent system for control/analysis that must be performed within a specified real-time window, in order to meet the demands of the given application. Timing is defined as responding to new data frames of 0.1 to 1.0 second intervals. Here we describe our efforts to develop two prototypes. Prime emphasis was on a controller expert system to show real-time performance in a cryogenic propellant loading application, and safety validation implementation of this system experimentally at the USAF Cape Canaveral Air Force Station Atlas Complex-36, using commercial-off-the-shelf software (cots) tools and object oriented programming (oop) techniques. This 'Smart GSE' (Ground Support Equipment) prototype is based in C, with imbedded expert system rules written in the CLIPS protocol. The relational database ORACLE® provides non-real-time data support.

The second demonstration develops the Vehicle/Ground Intelligent Automation concept, from Phase-I, to show cooperation between multiple expert systems (and conventional software modules). This 'Automated Test Conductor' (ATC) prototype utilizes a Knowledge-bus approach for intelligent information processing by use of virtual sensors and blackboards to solve complex problems. It incorporates distributed processing of real-time data and object-oriented techniques for command, configuration control, and auto-code generation.

BACKGROUND

The Air Force and NASA have recognized that our nation's current suite of launch vehicle systems has a number of problems making them inadequate for the projected needs after the late-1990's. A reduction in cost to \$300/lb of payload delivery, to LEO, 0.999+ reliability, high resiliency (elimination of long standdowns of many months), and high launch rate capacity are reasons behind the joint USAF/NASA effort for an operational ALS and Shuttle 'C'. ALS will serve the commercial and DoD mission models beginning in 2000. In order to meet the goals of \$300/lb and launch rates as high as 25 missions annually, on-board systems and their associated ground operations segment must be made as autonomous as possible, while at the same time improving reliability and safety. Under the ALS Program, a study was initiated to explore the use of EXPERT knowledge-based system (KBS) techniques for the purpose of automating the decision processes of these vehicles and all phases of the ground operations segment by assessing the feasibility, benefits, and risks involved.

An expert decision aid is a software approach to solving particular problems that are constantly changing over time and are complex or adaptive in behavior, the opposite of an analytical problem that is basically deterministic. Examples of these types of problems are: the re-scheduling of a vehicle checkout due to a damaged cable; or, determining if a system is indeed faulty given conflicting sensor readings. These heuristic problems require a depth of knowledge and experience (art rather than science) to form solutions quickly. Expert systems embody that collection of knowledge and experience in modular pieces that are rules and facts that describe the proper thought process for a given set of circumstances arrived at by any path. It is this modular independence that makes expert systems attractive. The incremental improvement of knowledge and experience can be built and tested readily without re-testing the rest of the software system, unlike conventional software that is difficult to maintain in a day to day changing environment [1].

PROJECT DESCRIPTION

The objective of this program is to develop, demonstrate, and evaluate the use of expert decision aids in areas that would improve ground and on-board system autonomy for the purpose of reducing the life cycle costs, shortening the processing critical path time, and improving safety and vehicle reliability. This technology program continued the work begun under the Phase-I study: Space Transportation Expert System Study (STRESS), contract managed by the USAF Wright Research and Development Center [2].

Tasks consists of an assessment of cost benefits vs implementation risks for specific applications, and demonstrations of key performance requirements to show feasibility within the selected application environment. Experience from our launch vehicle programs and other R&D

efforts shows that there are many opportunities in operations that reduce costs and improve autonomy, including:

- Ground operations: daily planning support and timely work-around decisions aids
- Ground checkout: autonomous operations and control
- On-board systems: monitoring, integration, and control
- Launch day: fly-with-fault diagnostics and decision aids

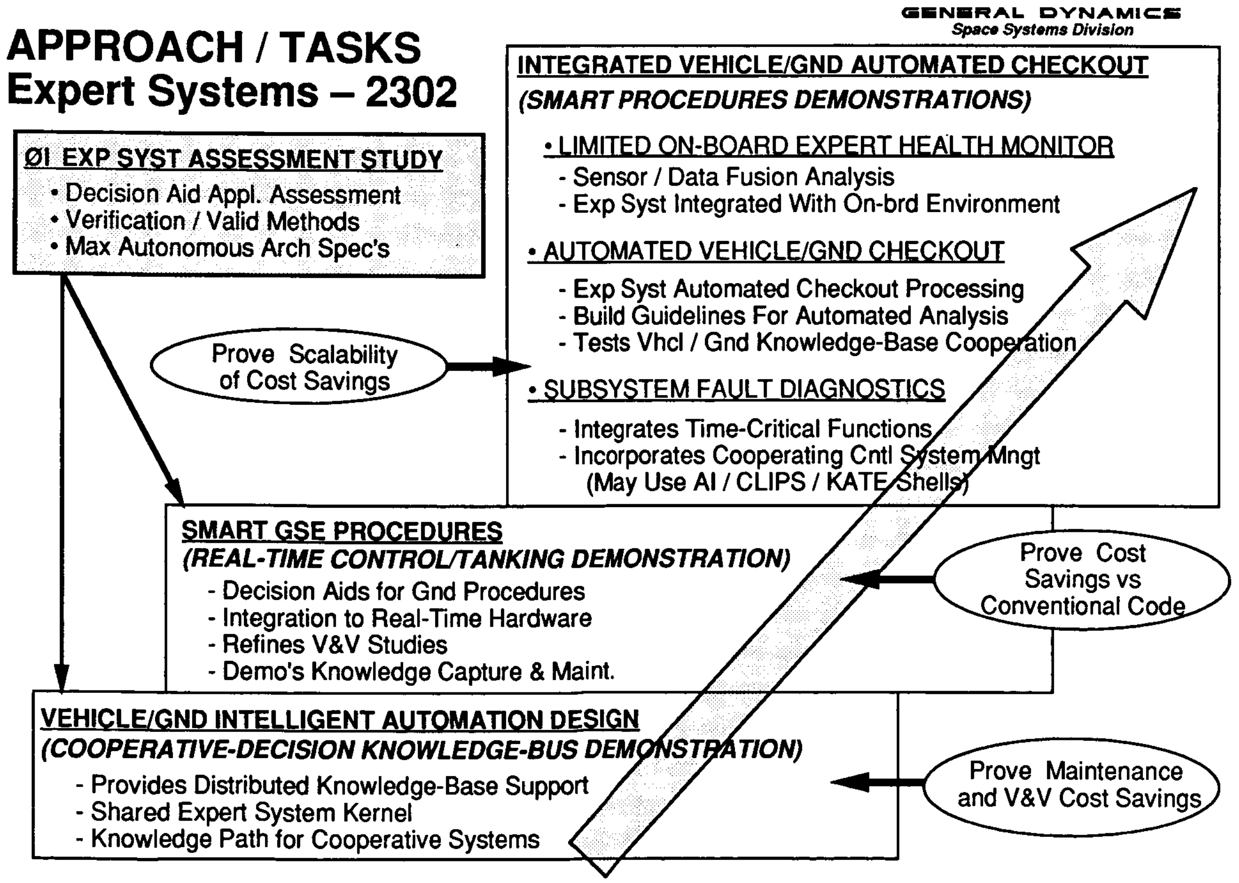
From 33 applications identified in Phase-I [3], cryogenic propellant tank loading was selected for a performance feasibility demonstration in order to reduce the risk of commitment to this developing technology. The Smart GSE prototype was of a fractional scale, sufficient to give a good performance correlation to a full-scale implementation. This demonstration intended to show:

- Ease of human interface to facilitate maintainability at the non-technical level;
- Real-time system performance for an appropriate level of complexity;
- Integration to both vehicle and ground hardware, and data systems;
- Validation methodology consistent for ground and on-board applications [4].

The second objective, targets the incorporation of KBSs into a combined Vehicle / Ground Intelligent Automation System. The "Knowledge-Bus" (K-bus) architecture, conceived in Phase-I, was used as the baseline concept in developing a maintainable mix of multiple conventional and knowledge-based systems. This layered architecture supports modularity and reusability of KBS components via object oriented programming techniques (hierarchial components, inheritance of methods, and polymorphism of functions) and knowledge-base partitioning [3] (for R/T performance and V&V efficiency). A second prototype demonstration of an Automated Test Conductor (ATC) system was begun to show cooperating intelligent systems in operation using blackboards and virtual sensors (elements of the K-bus concept). This approach is being successfully used to support the NASA Space Station Program. Elements of this architecture are already available in commercial-off-the-shelf software products. Development of an overall integrated architecture early in the investigation provides a context and focus for future demonstration prototypes, and assure the synergy of their gains in both development and use in vehicle operations, for example Integrated Health Monitoring (IHM).

This technical approach is shown in Figure—1.

SMART-GSE Procedures — This development task focused on supporting real-time control of time critical vehicle operations using KBSs. The application selection from the 33 candidates, the ranking shown in Figure—2, was a difficult choice. The demonstration had to require actual real-time servicing of decisions. It had to be sufficiently complex in scale to provide a good source of data for performance issues and for costs. It had to be non-trivial, i.e. dealing with real data so as to provide detailed feedback for correctness to serve as a benchmark for verification/validation testing. The final choice was cryogenic propellant loading of the Atlas launch vehicle. Although this application ranked in the middle, it satisfied all the requirements. The prototype would:


GENERAL DYNAMICS
Space Systems Division
APPROACH / TASKS
Expert Systems – 2302
Ø1 EXP SYST ASSESSMENT STUDY
• Decision Aid Appl. Assessment
• Verification / Valid Methods
• Max Autonomous Arch Spec's
INTEGRATED VEHICLE/GND AUTOMATED CHECKOUT
(SMART PROCEDURES DEMONSTRATIONS)
• LIMITED ON-BOARD EXPERT HEALTH MONITOR
- Sensor / Data Fusion Analysis
- Exp Syst Integrated With On-brd Environment
• AUTOMATED VEHICLE/GND CHECKOUT
- Exp Syst Automated Checkout Processing
- Build Guidelines For Automated Analysis
- Tests Vhcl / Gnd Knowledge-Base Cooperation
• SUBSYSTEM FAULT DIAGNOSTICS
- Integrates Time-Critical Functions
- Incorporates Cooperating Cntl System Mngt
(May Use AI / CLIPS / KATE Shells)
Prove Scalability of Cost Savings
SMART GSE PROCEDURES
(REAL-TIME CONTROL/TANKING DEMONSTRATION)
- Decision Aids for Gnd Procedures
- Integration to Real-Time Hardware
- Refines V&V Studies
- Demo's Knowledge Capture & Maint.
Prove Cost Savings vs Conventional Code
VEHICLE/GND INTELLIGENT AUTOMATION DESIGN
(COOPERATIVE-DECISION KNOWLEDGE-BUS DEMONSTRATION)
- Provides Distributed Knowledge-Base Support
- Shared Expert System Kernel
- Knowledge Path for Cooperative Systems
Prove Maintenance and V&V Cost Savings

Figure—1, Expert Systems ADP Approach

		CANDIDATE SYSTEM	Final Assessment
	22	Mission Planning with Automated Navig	38 49
	15	Pre-Flight Test Analysis	32 00
	26	System Wide Event Correlation	30 38
	28	Facilities Manager	29 34
	29	Mission Design Automation	29 27
	16	Post Flight Telemetry Data Analysis	28 72
●	27	Vehicle Test Conductor/Scheduler	27 15
	24	Command and Control Scheduler	26 78
	31	Payload Manifesting	23 81
	11	Vehicle Processing Logger System	23 05
	10	Operation Troubleshooting	22 15
	9	Launch Complex Environ Control System	19 48
	6	Integrated Test Ctr for Vehicle System	19 27
	5	Operator Training Simulator	19 14
●	19	Propellant Tanking of Vehicle	18 13
	18	Pneumatics, Press., and Purge Controls	17 03
	25	Support for the Decision to Launch	16 79
	32	Countdown Operations System Monitor	16 77
	23	Range Safety System	16 32
	8	Automatic Recorder Assignment	16 05
	3	Critical Parameter Vehicle Surveillance	15 61
	21	Guidance Calibration	15 25
	2	Limit Testing	14 85
	33	Telemetry/Landlines Checks & Assignm	14 57
	17	Flight Control Power Applic. and Monito	13 78
	30	Range Safety Sys. and Recovery Operatio	13 55
	7	Automatic Remote Sensor Calibration	13 37
	4	Hazardous Gas Identification and Safein	12 61
	20	Engine Ignition Ground Perform Mon	11 75
	12	In-Flight Engine Perform Monitor	8 03
	13	Fluids Analysis Health Monitoring	7 83
	14	Abort/Alternative Mission Modes (AGN&(	6 41
	1	Data Compression Analysis	5 58

Figure—2, Final Relative AI Candidate Ranking

- Prove R/T integration with GSE hardware and software by being installed at USAF Cape Canaveral Air Force Station Complex (CX)-36. Timing being defined in terms of 1 second decision loops based on monitored feedback
- Provide the V&V benchmark by using the validated Tanking Simulator available in the laboratory and later by using the pad validation equipment
- Provide comparative development cost data wrt the same task being done in conventional S/W on Titan/Centaur
- Provide long term maintenance cost data by being put into service at CX-36 and compared to Titan/Centaur

The basic approach to the Smart GSE system was a tanking controller built on a workstation using tools and standards so as to make it portable to any variety of computer systems. The demo used the CLIPS expert system shell from NASA/JSC. This shell had shown promise in our internal R&D efforts as capable of supporting real-time operations with suitable extensions.

We based all graphical interfaces on the X-window standard and Object Oriented Programming techniques. Here we used the Transportable Application Environment (TAE) provided by NASA/Goddard. This oop tool works on nearly all workstations and Macintosh systems under X. To provide the R/T feedback into the controller expert system, we used a PC version of our existing validated Tanking Simulator connected via a RS232 interface to the SUN workstation. We were using a SUN 3 system but planned to move the demo to a SUN 4 to do the timing tests. Later this would be moved to a Silicon Graphics workstation system for porting to CX-36.

Automated Test Conductor (ATC) — This development task focused on supporting the integration of KBSs into the vehicle processing environment; i.e. to actually have multiple expert systems cooperate in the performance of a given operation. The approach was to use the K-bus techniques in a judicious fashion to demonstrate the concept was do-able and without costly overhead making the concept impractical. The demonstration would prove that it is possible to have distributed knowledge-base support for control activities. This would open the door for accepting many of the 33 applications detailed in the Phase-I report. The driving force is the potential cost savings by having shared software kernels, object encapsulation of practices, procedures, and knowledge — to reduce validation, maintenance, and training. Further, the automation can now extend into the management of systems and not just isolated operations.

The basic approach in this task was to use multiple expert system modules, orchestrated by an ATC module, and running concurrently on 3 or more workstations. The workstations are initially networked via Ethernet TCP/IP protocol using socket transfer. For the R/T control management demonstration, we are in the process of installing VME hardware linkages between the workstations for message passing. The software being developed followed the principles of the K-bus. Throughout this project we used standard 'C' language and the GNU 'C++' oop language. For UNIS (Unified Network Information System) interfacing we used our own SQL-based data bridge to the ORACLE® relational data management system. This provided specification data to the expert systems upon demand. The initial test was a simple cooperative tanking task between subsystems. Later the Smart GSE controller becomes integrated in this encompassing system.

SMART-GSE DEMONSTRATION

The system level requirements were based on inputs from the Atlas CX-36 design and the system architecture developed under the ALS basic pre-design effort. Figures—3,4 show the required tasks and data-flows at the level of the Propellant Tanking Manager and one of the 7 subsystems. The primary features are first, the separation of management/control from health monitoring for a R/T system; and second the separation of R/T feedback monitoring from diagnosis.

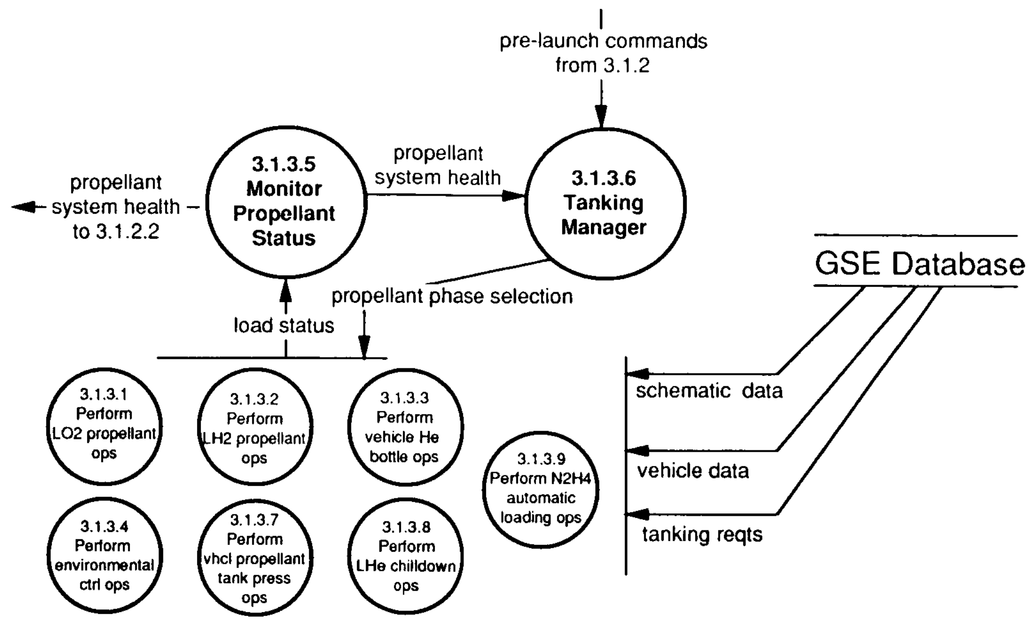
The prototype demonstration configuration is a laboratory closed-loop test linking the Tanking Simulator with the expert system controller. The Tanking Simulator consists of the existing Atlas GSE models with inputs from Ground Skid models and sequencing information from actual telemetry flight-test data. The Tanking Simulator is operated in a personal computer with a 100 millisecond cycle basis. This is connected via an RS232 line to a SUN 3 workstation running the expert system prototype. Preliminary validation tests were to have been done here. Later this system configuration would be modified to connect to the developmental launch control computers in San Diego for initial integration of the actual data telemetry streams. With this accomplished the transfer to CX-36 should be relatively straight forward. Testing at CX-36 would center on reading the telemetry data and determining performance variations in a variety of stressed situations. This testing would use the pad's validation equipment and not an actual vehicle. Timing test would be performed under a R/T Unix system on a Silicon Graphics workstation. Both the conventional equipment and the expert system would operate in parallel and comparisons of performance would be evaluated manually. The demonstration process is depicted in Figure—5.

The Smart GSE software configuration is shown in Figure—6 and highlights the significant elements. Some of these include the NASA/JSC: CLIPS expert system shell, the NASA/Goddard: TAE object oriented shell, and the ORACLE ® Relational Data Base System that was bridged to for specification type data as needed.

The primary emphasis in this project has been to demonstrate that expert systems can operate in real-time environments. It is

SMART GSE REQUIREMENTS (Level 3.1.3) Expert Systems — ADP 2302

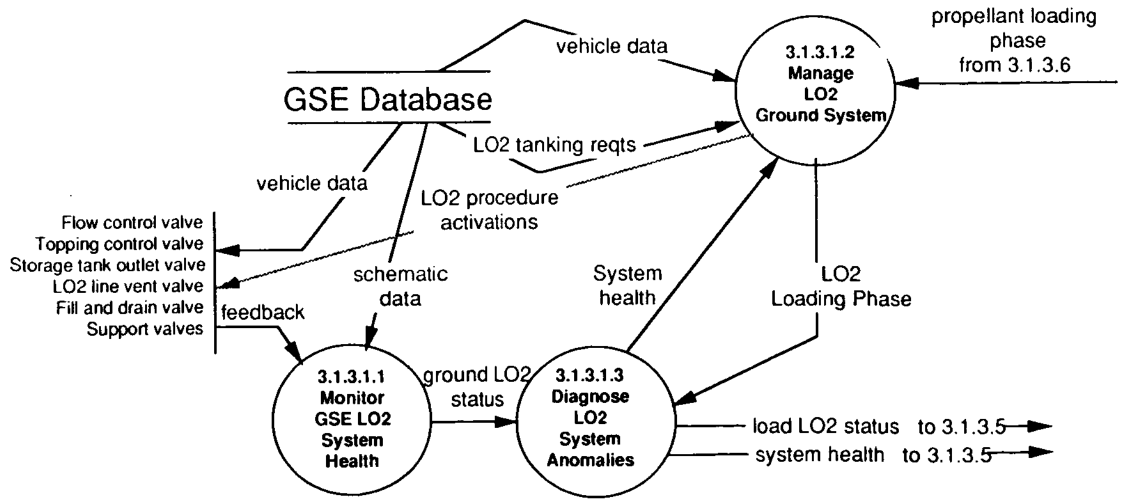
- Evaluated R/T Issues for Command & Cntl wrt ES
- Separated Mngt from Health Monitoring


pre-launch commands from 3.1.2
propellant system health to 3.1.2.2
3.1.3.5 Monitor Propellant Status
propellant system health
3.1.3.6 Tanking Manager
GSE Database
propellant phase selection
load status
3.1.3.1 Perform LO2 propellant ops
3.1.3.2 Perform LH2 propellant ops
3.1.3.3 Perform vehicle He bottle ops
3.1.3.9 Perform N2H4 automatic loading ops
3.1.3.4 Perform environmental ctrl ops
3.1.3.7 Perform vhcl propellant tank press ops
3.1.3.8 Perform LHe chilldown ops
schematic data
vehicle data
tanking reqts

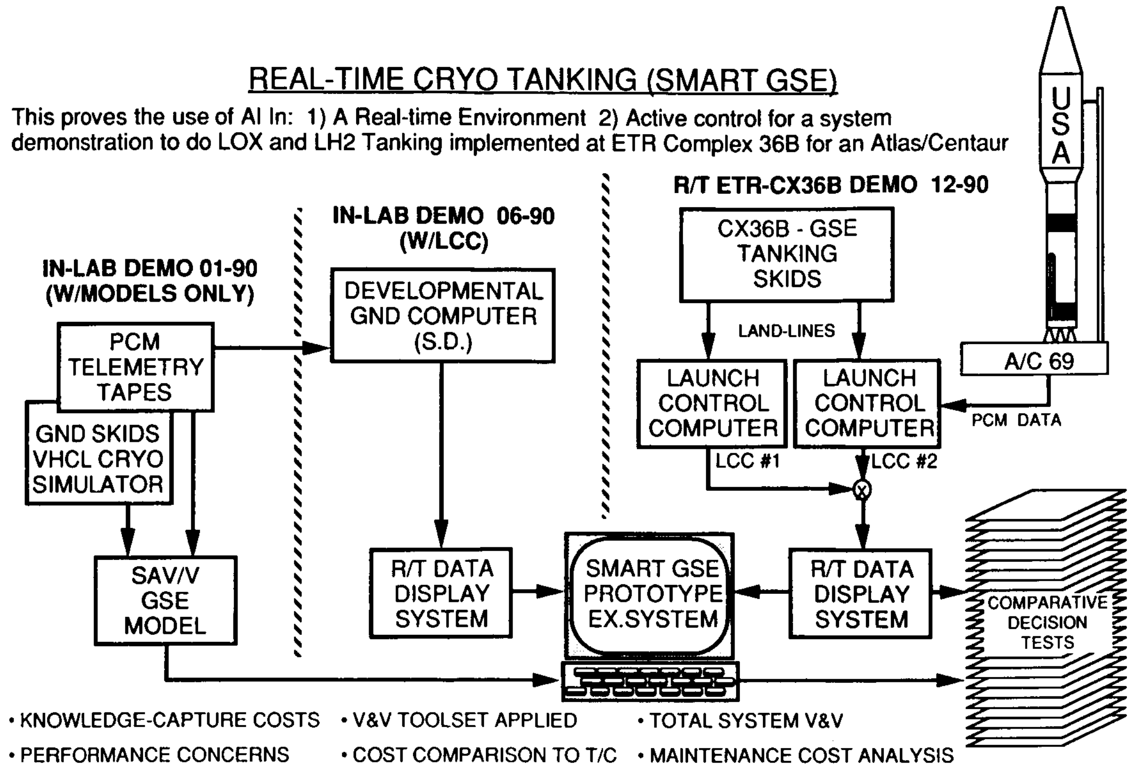
Figure—3, Propellant Manager requirements

SMART GSE RQTS - LO2 (Level 3.1.3.1) Expert Systems — ADP 2302

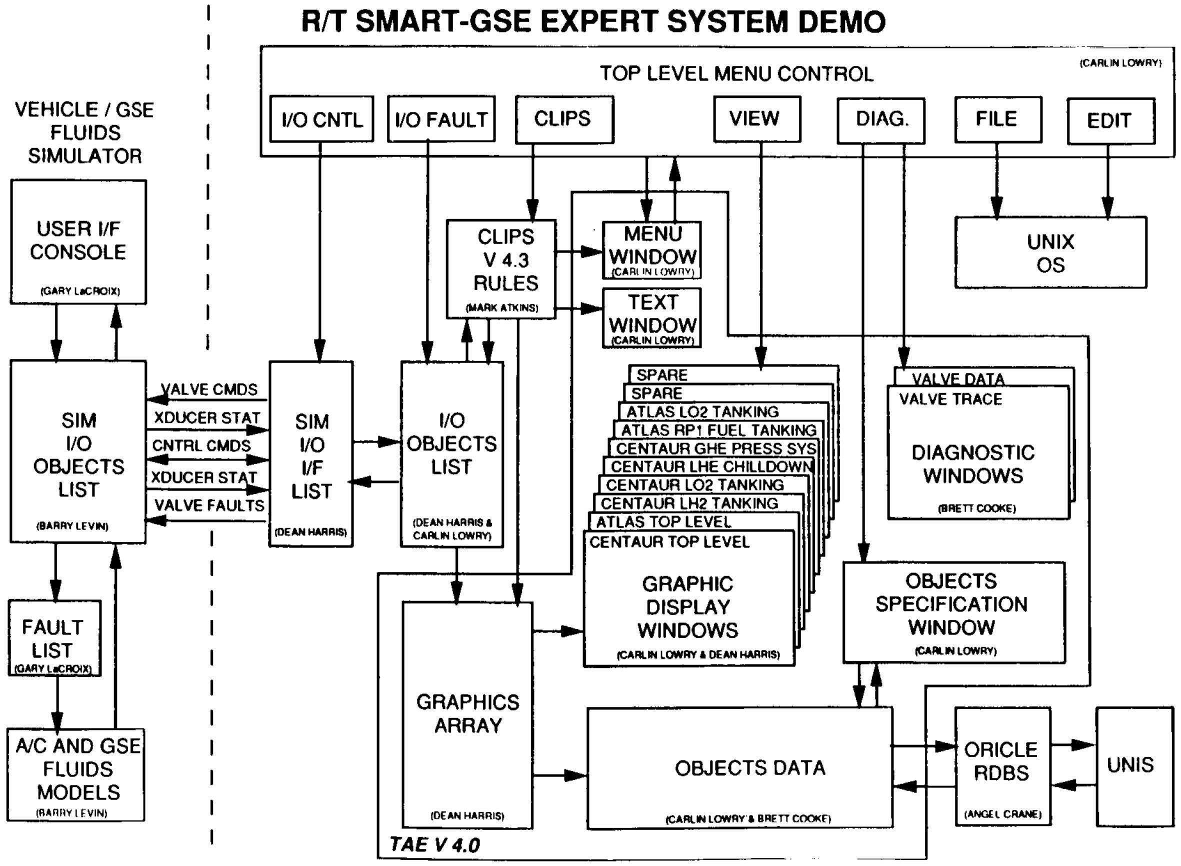
- Evaluated Knowledge-bus Issues wrt Command & Cntl
- Separated Monitoring from Diagnostics



Figure—4, LO2 Subsystem Manager requirements


REAL-TIME CRYO TANKING (SMART GSE)
This proves the use of AI In: 1) A Real-time Environment 2) Active control for a system demonstration to do LOX and LH2 Tanking implemented at ETR Complex 36B for an Atlas/Centaur
IN-LAB DEMO 01-90 (W/MODELS ONLY)
IN-LAB DEMO 06-90 (W/LCC)
R/T ETR-CX36B DEMO 12-90
USA
PCM TELEMETRY TAPES
GND SKIDS VHCL CRYO SIMULATOR
SAV/V GSE MODEL
DEVELOPMENTAL GND COMPUTER (S.D.)
R/T DATA DISPLAY SYSTEM
SMART GSE PROTOTYPE EX.SYSTEM
CX36B - GSE TANKING SKIDS
LAND-LINES
LAUNCH CONTROL COMPUTER
LAUNCH CONTROL COMPUTER
LCC #1
LCC #2
A/C 69
PCM DATA
R/T DATA DISPLAY SYSTEM
COMPARATIVE DECISION TESTS
• KNOWLEDGE-CAPTURE COSTS
• PERFORMANCE CONCERNS
• V&V TOOLSET APPLIED
• COST COMPARISON TO T/C
• TOTAL SYSTEM V&V
• MAINTENANCE COST ANALYSIS

Figure—5, Smart GSE shows evolution to full validation


R/T SMART-GSE EXPERT SYSTEM DEMO
TOP LEVEL MENU CONTROL
(CARLIN LOWRY)
I/O CNTL
I/O FAULT
CLIPS
VIEW
DIAG.
FILE
EDIT
VEHICLE / GSE FLUIDS SIMULATOR
USER I/F CONSOLE (GARY LaCROIX)
CLIPS V 4.3 RULES (MARK ATKINS)
MENU WINDOW (CARLIN LOWRY)
TEXT WINDOW (CARLIN LOWRY)
UNIX OS
SIM I/O OBJECTS LIST (BARRY LEVIN)
VALVE CMDS
XDUCER STAT
CNTRL CMDS
XDUCER STAT
VALVE FAULTS
SIM I/O I/F LIST (DEAN HARRIS)
I/O OBJECTS LIST (DEAN HARRIS & CARLIN LOWRY)
SPARE
SPARE
ATLAS LO2 TANKING
ATLAS RP1 FUEL TANKING
CENTAUR GHE PRESS SYS
CENTAUR LHE CHILLDOWN
CENTAUR LO2 TANKING
CENTAUR LH2 TANKING
ATLAS TOP LEVEL
CENTAUR TOP LEVEL
GRAPHIC DISPLAY WINDOWS (CARLIN LOWRY & DEAN HARRIS)
VALVE DATA
VALVE TRACE
DIAGNOSTIC WINDOWS (BRETT COOKE)
OBJECTS SPECIFICATION WINDOW (CARLIN LOWRY)
FAULT LIST (GARY LaCROIX)
A/C AND GSE FLUIDS MODELS (BARRY LEVIN)
GRAPHICS ARRAY (DEAN HARRIS)
OBJECTS DATA (CARLIN LOWRY & BRETT COOKE)
ORICLE RDBS (ANGEL CRANE)
UNIS
TAE V 4.0

Figure—6, Smart GSE prototype features include CLIPS, ORACLE, and TAE cots tools anticipated that the CLIPS shell would not be the only expert system shell used in the course of the Project development. Aspects of the Vehicle/Ground Intelligent Automation study required for test planing and procedure coordination may better use cots expert systems shells for goal oriented activities.

The prototype Smart GSE demonstration actually consists of several expert systems working together. These are presently modules linked within the same program, as apposed to being true cooperating distinct systems.

Propellant Manager Expert System — This ES is required to accept a set of primary commands from the Mission Manager and perform the necessary operations on the hardware systems to accomplish the functions that the he has selected. A sample list of the commands that the operator can select is:

TANK, DE-TANK, START, STOP, LAUNCH, HOLD, STATUS, ...

The Propellant Manager ES portion of the Smart GSE then performs any necessary correlation between the different systems under its control and issues a set of secondary commands that will insure that the safety of the vehicle, pad, and launch area will be maintained. Further, the Propellant Manager ES coordinates the data about each of the subsystems and adjusts each to ensure that they function smoothly between systems and prelaunch phases. The data or status at this level is also sent up to the Mission Manager for evaluation.

Subsystem Controller Expert Systems — The Propellant Manager coordinates commands for several Subsystem Controllers. These sub expert system Controllers are for each of the different fluid systems (LH2, LO2, LHe, Hydrozine, ...) and act upon the commands issued by performing the functions that have been requested.

These Controllers are concerned with the detailed functional commands to control the physical hardware and turn on or off specific sequences of valves or other equipment. Typical command decision-sequences made by the local Controller on the functional elements would be:

FAST FILL, TOP OFF, PRESSURIZE, DUMP, PURGE ...

The subsystem Controllers process function (object) commands by accessing a call to a set of conversion routines that would take the function (object) request and perform the necessary hardware interpellation. This is actually turn on/off the necessary valves or equipment to perform the function in question. An example sequence of valves would be:

(F-34-5 ON) (F-35-5 ON) (F-7-1 OFF) (F-22 OFF) ...

These Controllers also have responsibility for selecting the primary or backup functions as necessary.

Health Monitoring Expert System — A Health Monitoring expert system correlates feedback information and statuses the health of the equipment. All critical function feedback is sent to both this Health Monitoring expert system and the expert system Controller. This direct feedback allows fast close-loop response by the Controller. Non-critical anomalies are analyzed by the Health Monitor expert system. These problems are evaluated using the expert system data fusion capability to ascertain if a true failure has occurred. It then isolates the cause of the failure and notifies the expert system Controller of the defect. This allows the Controller to select appropriate backup functional equipment. In parallel, the Health Monitor expert system notifies the Propellant Manager Health Monitor of the defect and requests repairs. All anomalies are reported to this top level Propellant Manager Health Monitor to correlate with the other active systems.

Simulation training and validation testing — A monitoring / debug mode of operation will be available on the controller side of the sub level expert system to make suggestions of what operation the operator should be selecting and why. In the validation phase a 'scripting' capability would be used for dynamic system testing.

Progress to Date — The controller begins with a simplified schematic display of all nine propellant systems. A menu bar resident at all times allows the human controller to choose options from pull-down menus. These features include viewing alternate schematics, performing diagnostics on the imbedded expert systems, and manipulating the program parameters to explore or control specific scenarios. A more detailed view of any subsystem replaces a top level schematic when the pertinent name field is clicked on the top level schematic itself or when the name is chosen from a pulldown menu.

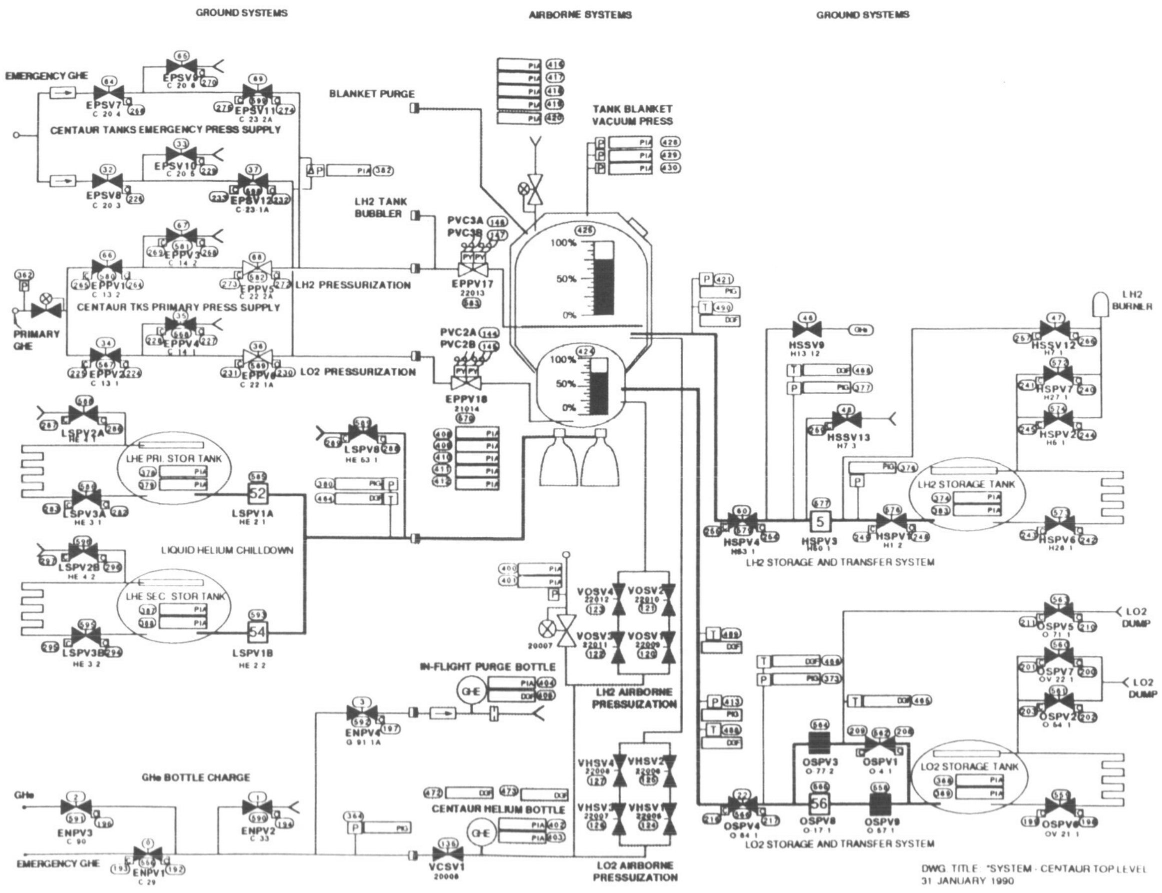
Each schematic consists of a background with overlaid discrete items. Valves, connections, and the tanks are discrete objects. The 'valve objects' change color to indicate changed state, such as open, closed, fault, or warning. A window containing both static and dynamic information about a schematic item is activated by clicking on the desired item. This information comes thru. the ORACLE® database bridge. A stretcher object reacts to the changes in the propellant flow and indicates the current tanking percentage completion on a sketch of the 'tank object'.

Figures—7 is the top functional level schematic; Figure—8 is a photograph from the SUN 3 display of the Smart GSE prototype. The photograph displayed is one of two top-level system displays, the one for the Centaur is shown and there is one for the Atlas vehicle/ground propellant systems. The remaining displays are sublayer displays of subsystems and show increased detail of all the operational components that facilitate the primary functions depicted on the top-level displays.

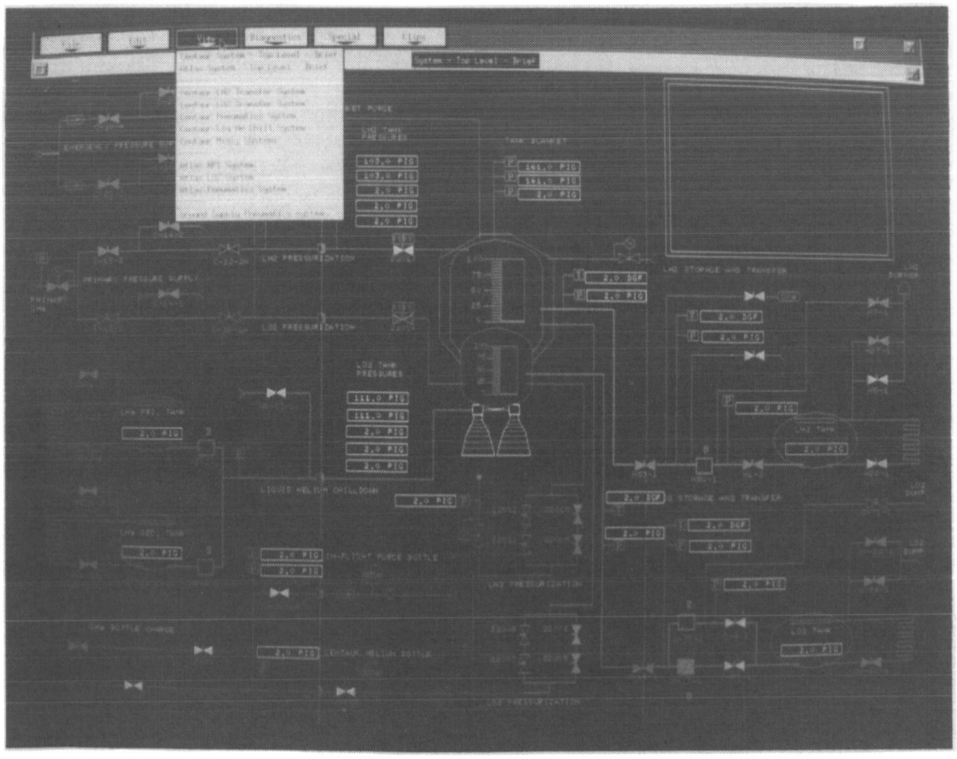
Operationally the ES knowledgebases for the LO2 and LH2 subsystems were completed and had begun initial integration testing. The displays were completed and had begun integration testing with the Tanking Simulator. The displays had not completed the animation software to show the actual closed-loop data feedback from the Tanking Simulator. The other subsystems knowledgebases were in various stages of development when work was terminated.

Real-Time Issues — The application as a controller implies control of the real-time decision process in relationship to the external environment that it deals with. Several methods were explored in an attempt to devise techniques for controlling the inferencing process and the rule-set that it evaluated. Some of these methods are shown in Figure—9. The use of SALIENCE is one of those techniques that is well suited to this area of expert systems. Simply put, SALIENCE is another term for rule-priority. How that priority is established, how the priority is maintained over time, and how the priority scheme interacts with the Inference Engine tie-breaking mechanism are all important application considerations [5].

The examination of rules that do not fire during an expert system application's cycle is the unfortunate overhead that expert systems typically carry. Another method used was to create a rule partitioning approach that would not require modification of the standard CLIPS shell (as tried in the Portable Inference Engine [6]), but would instead be application-specific CLIPS code that could reduce the number of rules present in the Rete Net at any given time. There are only a few expert system development tools that implement the concept of knowledge base rule 'clustering' where production rules are organized into logical arrangements to facilitate better control over their execution. An example of this approach



Figure—7, Top functional level SCHEMATIC for Centaur propellant systems



Figure—8, Top functional level OBJECT-ORIENTED DISPLAY (called 'Function Control Blocks') is found in IBM's Expert System Environment (ESE). The challenge was to create a similar rule partitioning approach in CLIPS without any significant degradation in performance - i.e. continual real-time operation. The method [1] removes rule 'clusters' that were not used during a real-time cycle and adds any 'clusters' that are required during that same cycle. In developing our CLIPS approach, minimum overhead costs were unavoidable. To date, R/T operational performance has not been tested.

Using this approach, verification and validation techniques for expert systems are more likely to succeed than with traditional expert systems. By efficiently 'clustering' the rules and facts, modular testing of modifications/enhancements are easier to perform than non-modularized expert systems applications — i.e. each partitioned rule 'cluster' can be independently verified and validated. This is a significant advantage over expert systems with a non-partitioned knowledge-base, and can ultimately lead to lower expert system maintenance/enhancement costs, and with better documentation.

AUTOMATED TEST CONDUCTOR (ATC) DEMO

The preparation and launch of a contemporary space vehicle is a labor-intensive process caused by the need for cooperation between many interdependent systems. In the last decade commercial software tools, capable of capturing the knowledge and practical experience of an expert as well as represent the design of the system, have been developed that can contributed to saving both cost and schedule. However, most of todays intelligent program packages suffer from the inability to communicate or act cooperatively with conventional systems or similar systems, or operate in the R/T environment. Our previous R&D experiences have explored the use of expert systems technology in practical situations. We have demonstrated expert systems operate in near real-time, work cooperatively with conventional R/T systems, work interactively with object-oriented graphics, and integrate with an off-the-shelf relational database management system. These R&D projects demonstrated the essential capabilities needed for cooperation between distinct intelligent modules, real-time communication, and intelligent access to stored data files.

Based upon the Vehicle/Ground Intelligent Automation concept from Phase-I, we are attempting to demonstrate cooperation between multiple expert systems (and conventional software modules). This 'Automated Test Conductor' (ATC) prototype utilizes a Knowledge-bus (K-bus) approach for intelligent information processing by use of virtual sensors and blackboards to solve complex problems. It incorporates distributed processing of real-time data and object-oriented techniques for command, configuration control, and auto-code generation. Figure—10 pictures the final goals of this demonstration. Ultimately the K-bus would link conventional launch processing software and a distributed collection of expert system modules of various types with the vehicle avionics having perhaps some on-board intelligence for integrated health monitoring. The benefits would be proof of the concept R/T operability, scaled cost comparisons, and a testbed for V&V.

Abacus Programming Corp. and the LinCom Corp. were participating subcontractors.

Development Philosophies — The ATC demonstration itself will be developed according to four general philosophies: autonomy, distributed control by modular expert systems, information transfer at a high level, and the use of intelligent databases.

System level philosophies:

CRITICAL RESPONSE ISSUES FOR R/T OPS
Expert Systems — ADP 2302

√ • **Saliences (prioritized rules)**
• **Priority Scheduling (dynamic saliences)**
• **Progressive Deepening**
• **Variable Precision Logic (depth of analysis wrt value of results = f(time))**
• **Decision Analytic Techniques (depth of search wrt value of results = f(time))**
• **R/T A* Search (best-first search)**
√ • **Bypass ES with Interrupt**
√ • **Interrupt Inserted Facts**
√ • **Single Valued Facts (2 fields vs 4 fields: (obj,val))**
√ • **Retract Seldom Used Facts**
√ • **Clause Ordering (RETE algorithm)**
√ • **Shorten Variable Names**
toolset • **V&V Trimming (remove redundant facts, subsumption)**
√ • **Data Preparation (scaling, thresholding, changes-only in data)**
• **Partitioning of the KB**
√ • **Faster Algorithms (OPS-83, PIE)**
√ • **Perform Math Functions Outside ES**
• **Use Alternate Paradigm (frame, model-based, etc.. as suits problem)**
√ • **Quadruples Facts (4 fields: (obj,attrib.,value,confidence))**

Figure—9, Critical Response issues considered for R/T operations

• Autonomy — The ALS environment will require that support systems be relatively autonomous and capable of independent decision-making. This will reduce the need for a standing army of engineers and ease the impact of the anticipated loss of older experienced personnel.

• Distributed Control — The test conductor will coordinate distributed controls for ground systems equipment (GSE) and vehicle avionics systems. The GSE is a distributed system, distributed both functionally and physically. Functions such as test procedures, fault diagnosis, or scheduling may be performed by separate expert systems. Physical distribution will be permitted, with the demonstration performing on distinct workstations .

• High-level Intelligent Information Transfers — The demo will use symbolic or language links which will allow the modules to share common knowledge, while performing their distinct functions. This sharing will be intelligent in that it will anticipate knowledge requirements by the modules, and supply them in compatible forms. It is anticipated that the real-world system will supply this information through common real-time or non-real-time lines.

• Intelligent Databases — Data records and files will be handled according to specific heuristics which take into account interconnections and dependencies. Smart schematics, for instance, would cascade externally-caused changes to all relevant data records.

• Hardware Independence — The program will be developed so as not to be limited to a particular hardware network or machine. It is anticipated that the applications will be written in different languages, on different machines, using different operating systems or inference engines. The final system hardware is unknown; therefore, the test conductor will accommodate these varying components, with minimal alteration.

• Real-time Performance — The software selected must satisfy real-time performance requirements, when required. These requirements will be driven by the necessary integration to hardware and real-time software systems, such as the avionics software on the vehicle.

Software philosophies:

• Language Standards — Standard high-level languages shall be employed, to simplify development and maintenance. The language of choice is ADA. Other languages will be accepted, only when driven by higher standards, i.e. shells, cots packages, etc.

• UNIX Commonality — The R/T UNIX operating system will be the system of choice for the development and functioning environment of the ATC. This will provide for distributed system commonality while supporting real-time. UNIX is a wide-spread workstation operating system, and provides many libraries and abilities which are necessary for the ATC, such as interprocess communication, etc.

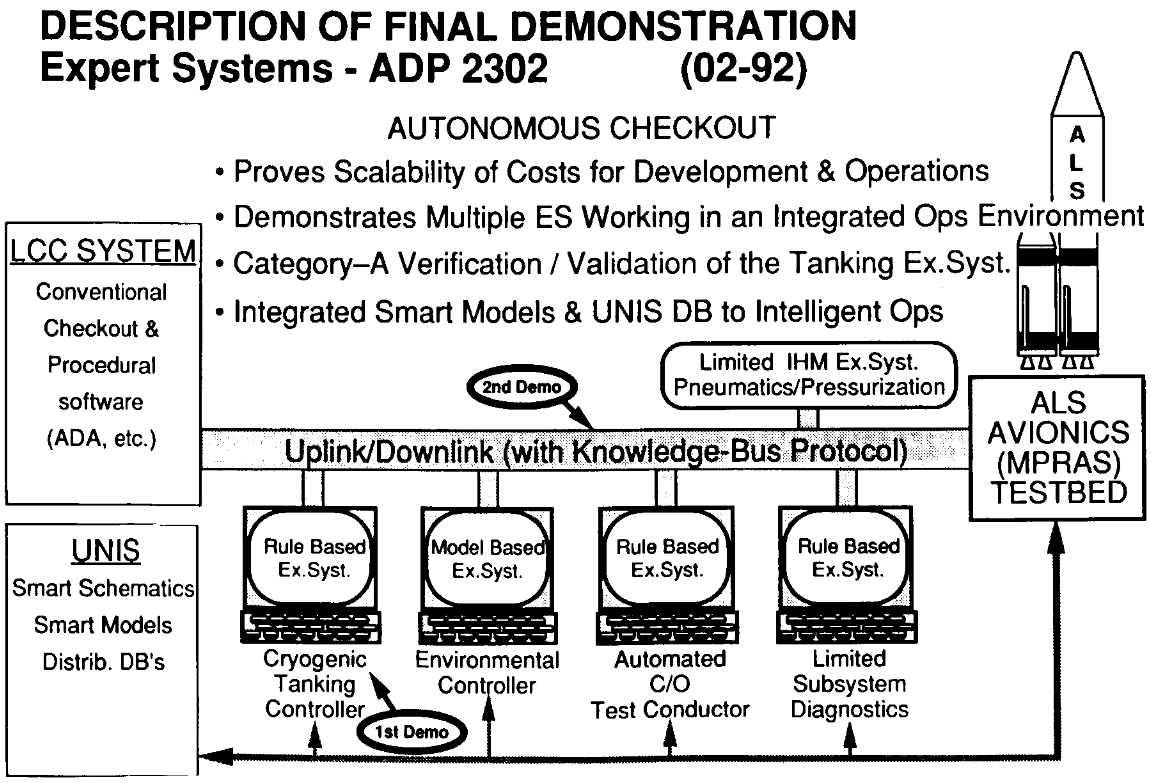
• Existing Toolkits — Commercial off-the-shelf (cots) products will be employed when they will adequately satisfy the task requirements. The benefits here are obvious.

• Object-Oriented Programming — The test conductor will be developed in accordance with object-oriented programming techniques. Currently the C++ language is targeted, promoting software reusability and functional modularity consistent with the object-oriented philosophy. TAE, a graphics package developed in C++ by NASA/Goddard, simplifies the use of XWindows, and promotes an iconic object-oriented user-interface.

• Shells and Environments — Commercial shells and environments that are proven to adequately satisfy the task requirements will be utilized. For the artificial intelligence portions of the testbed, such as the K-bus, CLIPS will be the shell of choice. CLIPS interfaces well with the C language and provides enough functionality to satisfy our R/T requirements. Other AI paradgms will be used as required.

• Permanent Data Storage — The ATC will need an imbedded database system, both for its system functions and its nodes. The database currently targeted is Oracle's relational database management system, which interfaces easily with the C language and is hosted on several workstations. A SQL-bridge will link the two.

Knowledge Bus — A critical component of the ATC demonstration is the Knowledge Bus. As its name implies, it provides a communication path for a higher level of information transfer, not simply data. The K-bus is a layered architecture in an object paradigm for development, integration and verification of distributed real-time systems, see Figure—11.


DESCRIPTION OF FINAL DEMONSTRATION
Expert Systems - ADP 2302 (02-92)
AUTONOMOUS CHECKOUT
• Proves Scalability of Costs for Development & Operations
• Demonstrates Multiple ES Working in an Integrated Ops Environment
• Category–A Verification / Validation of the Tanking Ex.Syst.
• Integrated Smart Models & UNIS DB to Intelligent Ops
A L S
LCC SYSTEM
Conventional Checkout & Procedural software (ADA, etc.)
Limited IHM Ex.Syst. Pneumatics/Pressurization
2nd Demo
ALS AVIONICS (MPRAS) TESTBED
Uplink/Downlink (with Knowledge-Bus Protocol)
UNIS
Smart Schematics
Smart Models
Distrib. DB's
Rule Based Ex.Syst.
Model Based Ex.Syst.
Rule Based Ex.Syst.
Rule Based Ex.Syst.
Cryogenic Tanking Controller
Environmental Controller
Automated C/O Test Conductor
Limited Subsystem Diagnostics
1st Demo

Figure—10, Autonomous vehicle checkout is possible using the K-bus approach

Systems implemented under the K-bus can include both knowledge-based and conventional procedural components. The K-bus features tools to ease development and coordinate functions that permit diverse applications to operate as a coherent system. Just as an operating system manages physical resources, the K-bus provides the means to access common reasoning services for embedded knowledge-based applications and analogous high-level services for procedural applications and V&V.

The K-bus concept was originally developed in Phase-I with a Design Specification for supporting a distributed network of cooperating expert systems serving an integrated Vehicle/Ground Mission Management System. It was anticipated that such a system would take the maximum advantage of semi-autonomous agent processes with knowledge-based communication and control to perform operations and vehicle/ground checkout. Phase-II applys these concepts to a working prototype:

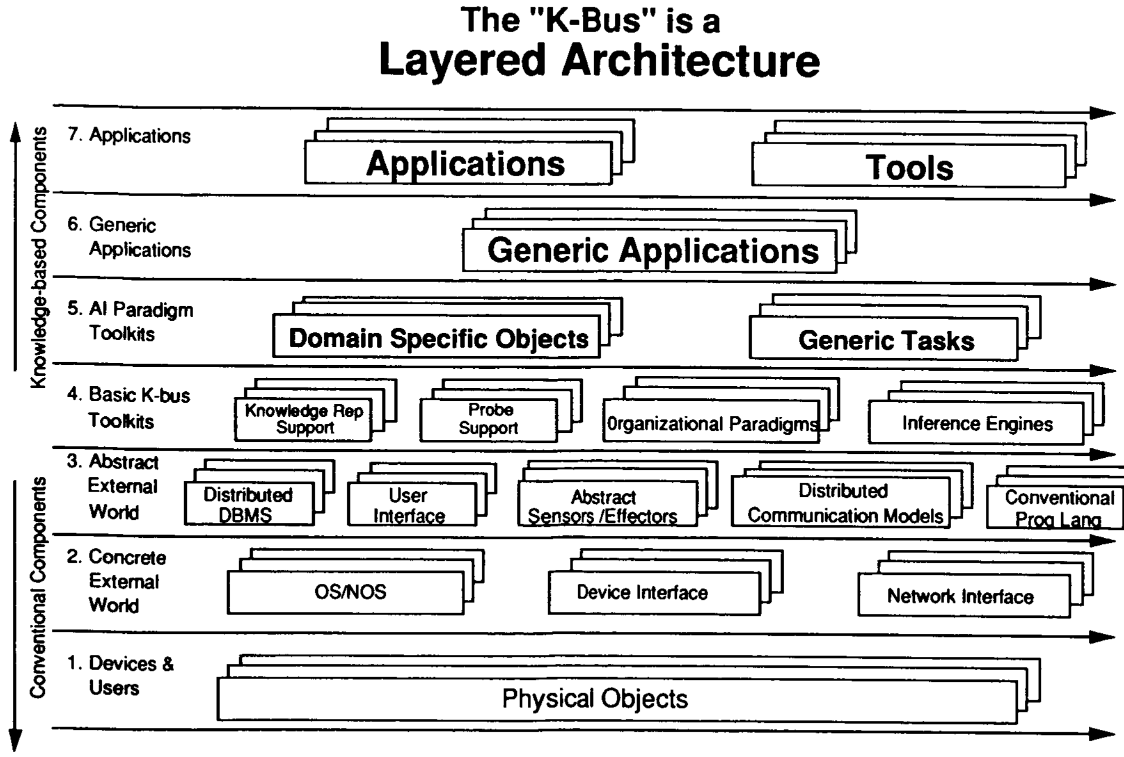
- Development of design specifications of the K-bus, and a User Manual describing how to use its facilities in developing a distributed application.
- Detailed design of the following K-bus objects: Object, SaveableObject, SystemCall, List, Buffer, String, Attribute, Socket, PostOffice, Finder, Message, KnowledgeUnit, KnowledgeSource, MessageManager, Agent.
- Implementation of these objects as a C++ library and Alpha testing of them with a simple driver program. This library was developed with Oasys C++ compiler and the Apple AU/X operating system.
- Rehosting the library from the development Oasys system to the Gnu C++ compiler and SUN workstation for integrated testing.

Aspects of the K-bus — As previously stated, the K-bus follows the distributed object oriented model of interaction between software modules, defined here to be loosely-coupled 'agents'. Further, this supports the open, continuous processing characteristic of cooperative systems, and which makes them much more complex than traditional consultation-based programs. This event-driven programming methodology is also shared by several conventional systems, such as X Windows. In each case, procedures ("event handlers") are associated with events that can occur asynchronously, such as the user clicking a mouse, or a database update.

Agent — The agent is the fundamental active entity in the K-bus, encapsulated as an object which communicates by messages. Currently an agent and its message manager occupy a Unix process, so its boundary exists not only as a software object but is also enforced at the operating system level. An agent is defined as a collection of knowledge sources and an organization. These knowledge sources may be implemented as expert systems or a conventional system. Each knowledge source has a list of capabilities and interests. This list matches questions it can answer and information it would like to be told. The agent advertises these attributes with the Finder and keeps a cache of other agents' capabilities and interests for subsequent communication.

An agent's specification thus permits implementation along several sizes of granularity. Internally, it can be a whole organization of problem solvers, or just a simple C program. It has a scheduler component for control of its knowledge sources and is not necessarily serial. Its state may be dormant or active, but currently most agents are eternally vigilant or waiting for a reply. For efficiency reasons in Unix-like environments a large grain may be preferred, and this can be used at the next layer up as a generic task (an agent which is a specialist in one area of problem solving).

An agent's capabilities and interests represent a model of its goals, plans, abilities and needs that other agents can use for



Figure—11, Autonomous vehicle checkout is possible using the K-bus approach cooperation. An agent can choose not to cooperate by not advertising this model, but in general they can build up more extensive models of each other by starting with the originally advertised capabilities and interests and then learning from experience by caching results. For example, two agents may have a capability to do arithmetic, but by trying each the faster one is identified and will be preferred in future requests. An agent can have a reflective ability by installing probes in itself (for example, to measure the number of rules fired by a knowledge source's inference engine). This ability allows it to monitor its progress and interrupt if necessary. The combination of agents into a cohesive problem-solving team is achieved by creating an organization. Figure—12 is an example of the internal organization of a complex agent.

Post Office – Each agent has a Post Office object, which queues incoming messages and permits addressing by name, rather than location. The Post Office uses a distributed Finder object, which keeps track of the addresses of active objects and maps them to their globally unique names. Furthermore, agents can advertise certain attributes (see later section) which are also registered with the Finder and permit communication by semantics rather than just syntactic names.

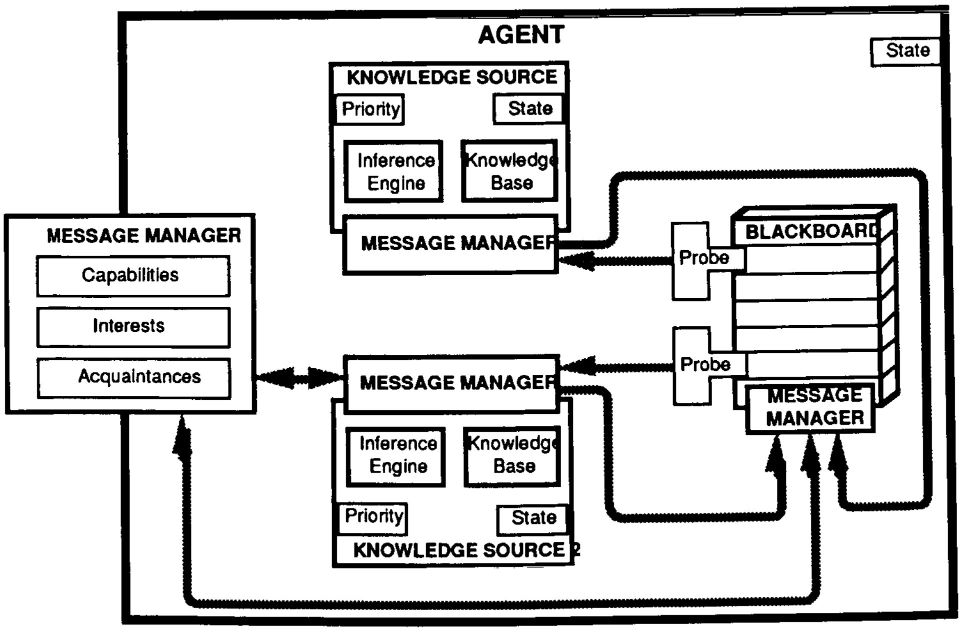
Message — The interaction medium is the message, the glue which enables the transfer of data and control between the agents. A message contains fields which identify the sender and receiver, an object (such as a question or answer) an optional time tag and list of attributes (which may include its expiration date or application-specific information). Control is passed by messages which represent remote procedure calls - they are intercepted by an agent's message manager. The message manager is responsible for converting messages to procedures and keeps a queue of questions received together with their askers (for subsequent direction of replies). Remote procedure calls by default are asynchronous (the caller doesn't block and wait for its completion), but may be synchronous if required (easier to program as it fits the conventional procedural language model). The question of whether the receiving agent blocks until it processes the request depends on the organization used: if the agent does - it is under the control of the sender (a client-server relationship); if not - it is autonomous. Of course, requests to lower-level services (such as a database manager) are processed synchronously - only high-level agents can own a thread of control.

Organization — An organization is simply a collection of agents who know each others' capabilities and interests, this is an implicit specification, encapsulated by knowledge existing in each agent. In contrast to structural definitions of organizations, this model is adaptive, since agents can compute who knows how to answer a question. New relationships can form within the organization. One agent can be programmed to act as a manager, delegating work to other agents according to their advertised capabilities, monitors their progress using probes and adjusts their position in the organization accordingly.

A method to combine agents more indirectly is by sharing access to a blackboard.

Blackboard — A blackboard is realized in the K-bus as a restricted subclass of agent - it is a passive server which is interested in everything (or at least whatever it is programmed for). Agents post information on the blackboard by sending it messages, they install probes on it to gather information resulting from matching events plus several current and historical conditions. A blackboard is thus a semi-permanent communication space, but also acts as a mechanism for a loosely-coupled organization whereby several agents can combine partial results without repeated inter-agent communication. It is more than a global database, in that the probes' histories provide a short-term memory and record of partial matches, so that new additions and requests can be processed quickly (in the style of the Rete algorithm for rule-based systems); in contrast, database queries are processed one at a time. This is an object-oriented version of the blackboard concept, and it is important to contrast it with blackboard systems which contain a centralized scheduler in control of the serial execution of agents - in the K-bus the agents are autonomous, and questions of parallelism and interference are answered by the message-passing architecture.

The blackboard's internal structure may be partitioned, to allow a for hierarchy of spaces available to groups of agents, but the external interface is ignorant of the internal structure of objects posted on it. Although logically centralized, it may be



Figure—12, An example of a complex Agent composed of several objects physically distributed for performance reasons - in this case, consistency must be maintained using techniques (e.g. multiple copies, deadlock avoidance) borrowed from distributed databases. A blackboard is demonstrated in Figure—13.

Interim Demonstration Configuration — The present ATC prototype was developed as an early assessment of the difficulties involved in operating a network of cooperating expert systems. The demo system is comprised of five expert systems (ES). Primary control is represented by the Automated Test Conductor (ATC) user interface. A message router, equivalent to the Finder mentioned above, handles information flow and command/control. The interface to the ATC is via an oop iconic display window, much like a Macintosh. This interim demo emphasized the concepts of distributed control and oop communication. Via a UNIS-like data management interface, oop scripts can be developed for a test scenario. Process ES objects may be assigned to any workstation or mainframe in the network and given initialization information. All results flow back to the ACT operator window. Each of the distributed sub-processes open their own respective windows on their hosted machine for inspection. This entire process is stored in object form and when initiated the software is auto-code generated, distributed, and executed. Figure—14 shows the five expert system configuration. The ATC user interface is in Figure—15.

REFERENCES

[1] Szatkowski, Dr. G., 'KB Partitioning Design for CLIPS', CLIPS Users Conference, Houston Tx., 1990.

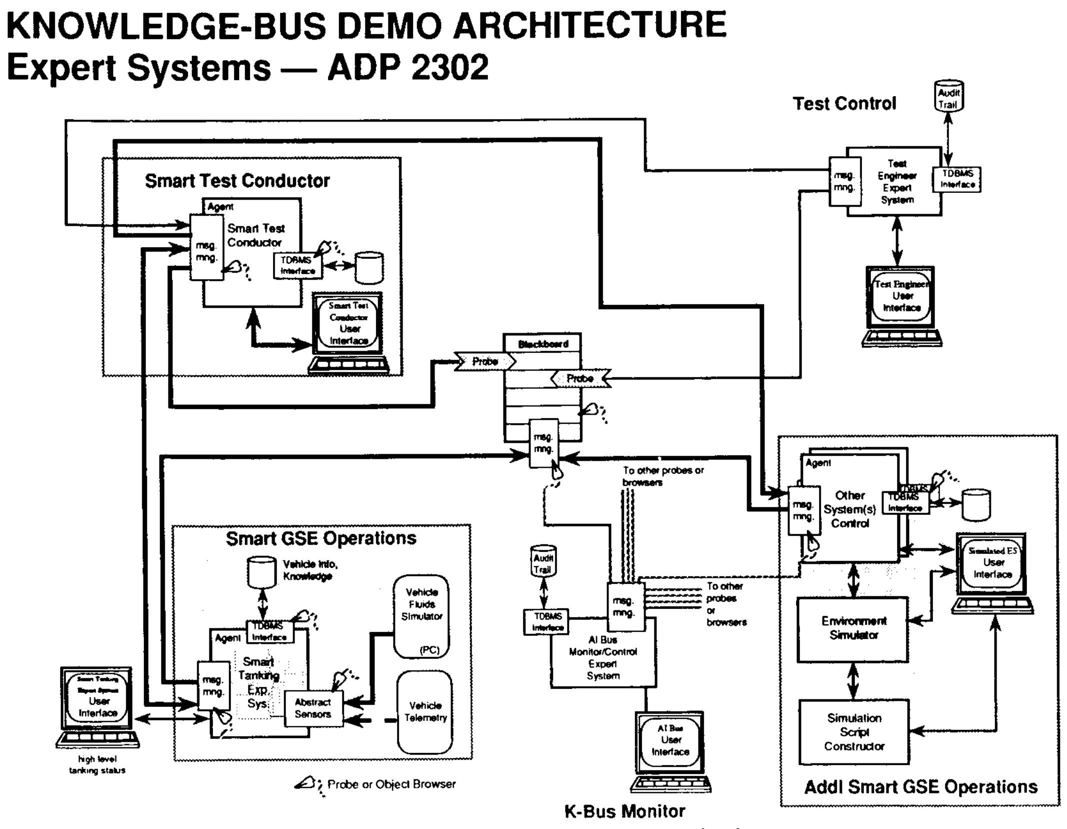
[2] Szatkowski, Dr. G., 'AI Decision Support for Low Cost Launch Vehicle Integrated Mission Operations', SOAR, Dayton Oh., 1988.

[3] Szatkowski, Dr. G., 'ALS STRESS Final Report', USAF contract F33615-87-C-3620, 1990.

[4] Szatkowski, Dr. G., 'Approaches to V&V of Imbedded Decision Support Systems Applied to Launch Vehicle Ops', Validation and Testing KB Systems Workshop, IJCAI, 1989.

[5] Szatkowski, Dr. G., 'ALS Expert Systems ADP-2302 Final Report', USAF contract F304701-88-C-0110, 1990.

[6] Le T. and Homerier P.,'Portable Inference Engine, PIE', Aerospace Corporation, LA Ca.

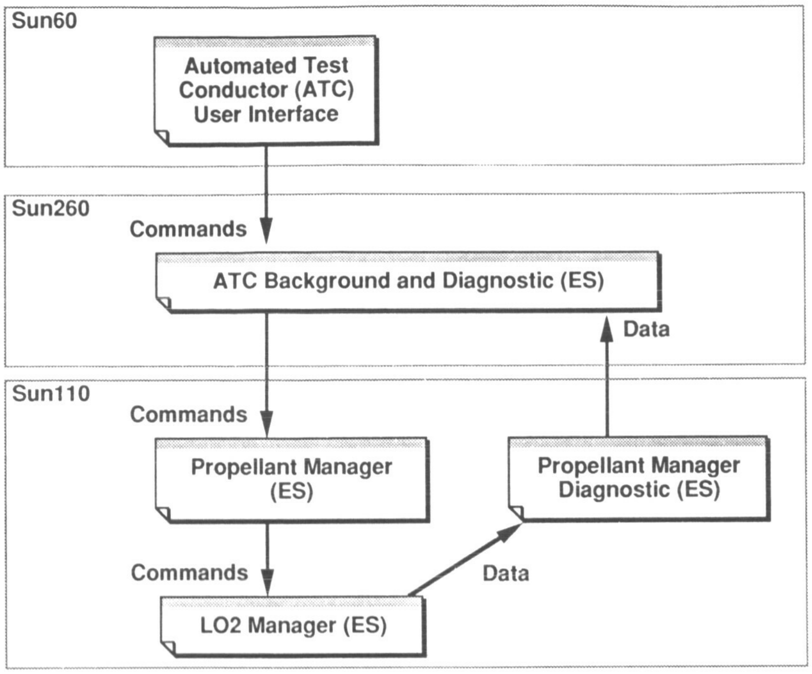


Figure—13, An example of a blackboard application

ORIGINAL PAGE IS OF POOR QUALITY

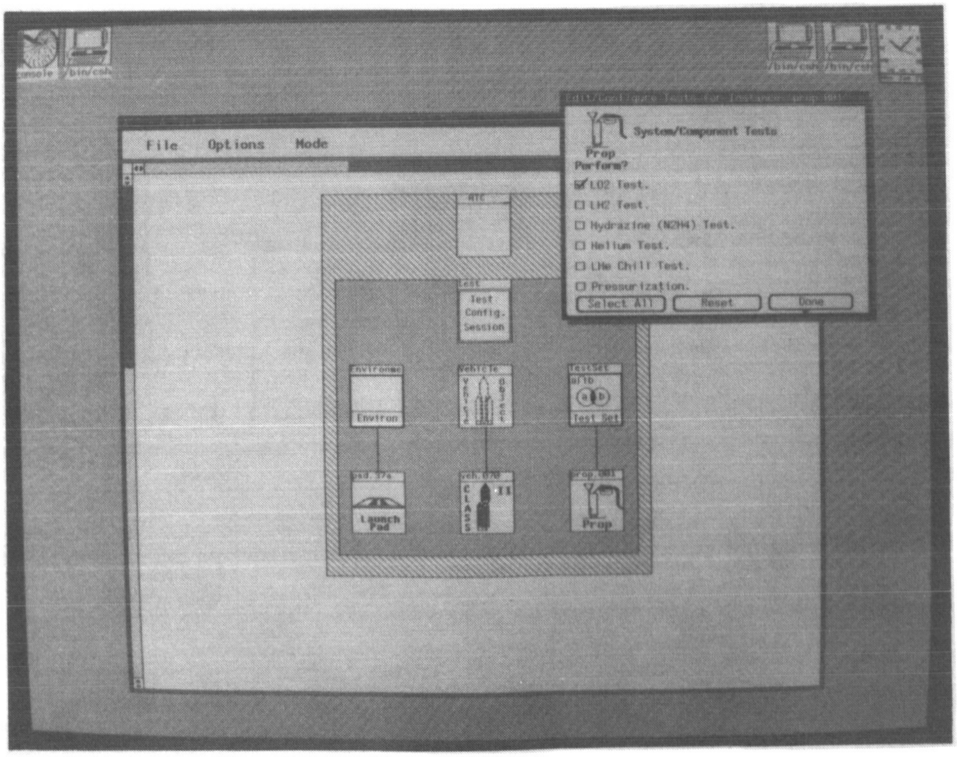
ADP 2302 - INTERIM DEMO
Vehicle Ground Intelligent Automation

Command, Control, and Data Flow



Sun60
Automated Test Conductor (ATC) User Interface
Sun260
Commands
ATC Background and Diagnostic (ES)
Data
Sun110
Commands
Propellant Manager (ES)
Propellant Manager Diagnostic (ES)
Commands
Data
LO2 Manager (ES)

Figure—14, ATC prototype demonstrated distributed multiple-Expert System cooperation



System/Component Tests
Prop
Perform?
- [✓] LO2 Test.
- [ ] LH2 Test.
- [ ] Hydrazine (N2H4) Test.
- [ ] Helium Test.
- [ ] LHe Chill Test.
- [ ] Pressurization.

Figure—15, Photograph of the ATC object-oriented Scripting capability

N91-20680

KSC'S WORK FLOW ASSISTANT

John Wilkinson and Earl Johnson
Kennedy Space Center
Lockheed Space Operations Company
M.S. LSO-002
Titusville, Florida 32780

ABSTRACT

The Work Flow Assistant (WFA) is an advanced technology project under the Shuttle Processing Data Management System (SPDMS) at Kennedy Space Center (KSC). It will be utilized for short range scheduling, controlling work flow on the floor, and providing near real-time status for all major Space Transportation Systems (STS) work centers at KSC. It will increase personnel and STS safety and improve productivity through deeper active scheduling that includes tracking and correlation of STS and Ground Support Equipment (GSE) configuration and work. It will also provide greater accessibility to this data. WFA defines a standards concept for scheduling data which permit both commercial off-the-shelf (COTS) scheduling tools and WFA developed applications to be reused. WFA will utilize industry standard languages and workstations to achieve a scalable, adaptable, and portable architecture which may be used at other sites.

1. INTRODUCTION

The task of preparing the Space Shuttle for flight is complex and involves systematically moving the shuttle through a series of work centers. The preparation activities at each work center involves coordination of a sizeable team of personnel and materials as well as the cumulative execution of some 25,000 operations per shuttle flow. While this coordination effort is scheduled daily in advance based on planned work and durations, the actual work required may vary by as much as 40% due to non-recurring activities revealed by scheduled inspections, e.g. correcting tile damage incurred during the previous mission. The limitation is that re-scheduling is done as a one day delayed reaction reconciling differences between planned and actual schedules.

In response to this need, the Work Flow Assistant (WFA) project of Kennedy Space Center's Shuttle Processing Data Management System (SPDMS) was initiated. This system is envisioned as a knowledge-based scheduling assistant acting primarily at the work center level. WFA will include the capability to: track schedule progress, the STS and ground support equipment (GSE) configuration and critical path on a near-real-time basis; and screen all work for compatibility to configuration. It will also provide the active scheduling and resource balancing of open standard work and for integration of non-recurring work into the schedule.

2. STRUCTURE AND FUNCTION

Functional aspects of the WFA will include:

- Information collection, assimilation, and dissemination.
- Multi-perspective user interface.
- Electronic signature capability.
- Tracking task events (e.g. enable, cancel, start, stop, hold, and resume).
- Dynamic critical path projection.

- Interactive queries
  - Status (e.g. configuration, activities in progress, etc.).
  - What-if schedule assessments.
  - Meta task event inquires (e.g. area close-out).
- Incorporation of non-recurring or non-standard work into scheduled activities.
- Proposed scheduling data exchange format

Figure 1 presents the top level logical organization of the WFA. Planning and scheduling, ranging from multi-mission 5 year planning to work center work package release for the next 11 days, will be accomplished through an integrated set of SPDMS host based applications . WFA will then construct and maintain a more in depth schedule taking individual resources and work center GSE configuration into consideration. WFA will receive transactions to start, stop, hold, resume, cancel and complete work. WFA uses this data to verify that prerequisite tasks are complete and that the current configuration is compatible before allowing work to begin. WFA also tracks schedule progress and configuration as the tasks are received; and maintains schedule data, critical paths and configuration in near real-time. WFA assists floor supervisors in the work centers to balance work and resources and to gain increased visibility into work center status. It assists flow managers and controllers of multi-work-center resources , e.g. quality or safety personnel, on a flow or multi-flow level to track and manage tasks in their purview. Near real-time status is also made available throughout KSC to keep the NASA and contractor work force informed on the progress of STS testing and the effect on their respective organizations.

3. CENTRAL CONCEPTS

The following presents the hinge-pins upon which WFA is founded. The approach to standards and modularity takes on unusual importance in that it determines whether WFA will be a single-use application or a reusable tool. Also information distribution and user customization concepts will heavily affect how well the many and varied user groups are served.

3.1 Standards

Like all major government installations, KSC has workstations, computer equipment and software from many vendors. Data paths required by the WFA span multiple vendor platforms and software packages. Standards utilization will lessen the implementation cost and have a positive effect on the extent of reusability. Using standards as the platform, rather than a single vendor model or line, permits maximum use of existing equipment and allows new acquisitions to be based on benchmark and cost considerations uniquely for each requirement.

Where no formal standard exists, ad hoc standards supported by multiple vendors are preferred.

Candidate multi-vendor standards or emerging standards are available which cover most areas of concern. One notable exception, for WFA's purpose, is the absence of standards governing the exchange of scheduling data between COTS scheduling applications. The existence of such a "standard", albeit ad hoc and limited in agency scope, will reduce the need for further custom developed software tools for WFA, and permit WFA technology to be more easily reused.

Analogous to the definition of an Application Portability Profile for Posix by the NIST, WFA development will include a set of APP tests and benchmarks to validate vendor independence and vendor supportability of a WFA release.

3.1.1 Industry and Adhoc Standards

When the multi-vendor support requirement is added to standards selection, the standards picture changes significantly. But without multi-vendor support, there is in effect no standard. While project approval of selected standards has not yet been secured, adequate candidate "standards" appear available for the following project roles: platform equivalent configurations, operating system, network connectivity, database, languages and graphical user interface.

3.1.2 Schedule Information Interchange Format

A unifying concept of the WFA is our proposed Scheduling Information Interchange Format (SIIF), which provides a logical common communications medium for transmitting and receiving scheduling information (Figure 1). The

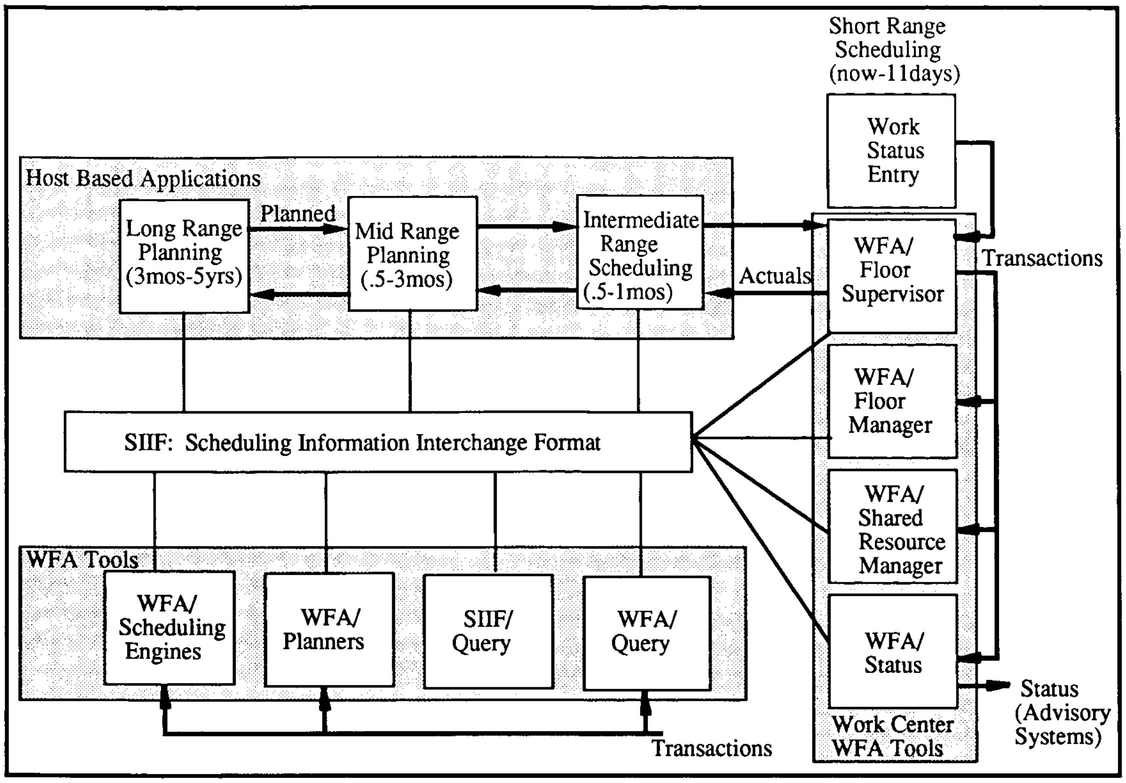

Short Range Scheduling (now-11days)
Work Status Entry
Host Based Applications
Long Range Planning (3mos-5yrs)
Planned
Mid Range Planning (.5-3mos)
Intermediate Range Scheduling (.5-1mos)
Actuals
WFA/ Floor Supervisor
Transactions
WFA/ Floor Manager
SIIF: Scheduling Information Interchange Format
WFA/ Shared Resource Manager
WFA Tools
WFA/ Scheduling Engines
WFA/ Planners
SIIF/ Query
WFA/ Query
WFA/ Status
Status (Advisory Systems)
Work Center WFA Tools
Transactions

Figure 1: Work Flow Assistant Logical Organization

SIIF defines a standard method for storing scheduling information in an SQL database form, as well as interface protocols for requesting services for each class of scheduling tool. Custom applications requiring SIIF scheduling services follow the SIIF protocol for requesting services from any SIIF scheduling tool.

The primary benefit is to make scheduling data readily accessible and scheduling tools interchangeable. Standardization of data is the next logical step for standards. After all, it is the data itself upon which decisions are made.

In order to make the SIIF practical, all participating data must be stored in SIIF rather than in the internal format of a particular tool. Also, each participating tool must be interfaced to the SIIF and participating applications must utilize the SIIF protocol. Naturally, this is not an all or nothing affair. Benefits will accrue proportional to usage. SIIF definition and compliance represents a significant decision, not yet made, that will require careful deliberation and active commitment.

3.2 Mix N'Match Components

The SIIF will be used analogously to a computer backplane for disseminating scheduling information into which scheduling tools and scheduling application can be installed (Figure 1). Each tool or application is installed via an interface program that maps the protocol and format of that product to the protocol and format of the SIIF. Tools are envisioned to be installed by functional category and to appear to the SIIF as having the same functional interface. as other tools in the same category. New COTS scheduling tools can then be installed by developing or modifying an existing interface program. Scheduling applications can use the installed tools as building blocks. Users can utilize any tool of a tool category to access any of the data maintained in SIIF format.

WFA scheduling engines are tools of special importance. They have a high level of functionality permitting them to be used by most scheduling applications and are scalable permitting hardware platform performance to be selected based on performance requirements.

3.3 Active Versus Reactive Scheduling

The traditional scheduling approach is reactive; that is schedule, do the work, periodically review progress versus the schedule, and then reschedule to resolve discrepancies . In contrast, WFA will receive task progress transactions (start, hold, complete, resource assignment, etc.) electronically and immediately update schedules and configuration status for each work center. Critical paths and individual resource utilization will also be tracked in near real-time. WFA will also assist floor supervisors, responsible for controlling work flow through a work center, remain abreast of work status, and in balancing resources and ordering tasks to improve productivity and minimize schedule impacts.

3.4 Configuration Based Scheduling and Safety

Scheduling and tracking at the work center level will utilize knowledge of the physical aspects and current configuration of the work center and the STS elements being processed as related to the tasks being performed. Configuration data includes considerations such as Orbiter power status, position of test stands and STS flight moveable surfaces, payload bay door configuration, etc. With each task, any configuration requirements or prerequisites must also be identified by appropriate organizations. Using this information, WFA schedules the work by taking configuration into consideration. Current configuration state will be gathered through manual entry, completion of a scheduling task for which a configuration effect is identified, or by processing STS and GSE real-time measurement data received from the firing rooms at KSC. WFA also tracks the current configuration and verifies that configuration requirements match configuration computed state and that all predecessor tasks are completed before permitting work to commence. While this method is not fool proof, it should significantly improve safety to the work crew and the STS.

3.5 Information Distribution

Three levels of information distribution service are provided by WFA. These services disseminate near real-time status from work centers supported by WFA. The three distribution levels are: assistants and direct support; interactive query and reports; and advisory systems.

Assistants and direct support serve the personnel actively working in and around a work center to accomplish work. Assistants are planned to act as intelligent aids to flow-strategic personnel including floor supervisors, planners and flow managers.

Interactive query and reports are provided to meet special user group needs for access to schedule and status information.

Advisory systems provide work and configuration data in a broadcast mode to user workstations that are primarily IBM PC compatible and utilize a set of KSC developed tools supporting user tailored display of that information. This distribution service makes flow and configuration information available efficiently throughout KSC in a low overhead computational manner and can be disseminated to other centers as well. A number of other advisory systems are being constructed at KSC that work in a similar manner using the same resources and technology.

3.6 Multi-perspective Malleable User Interface

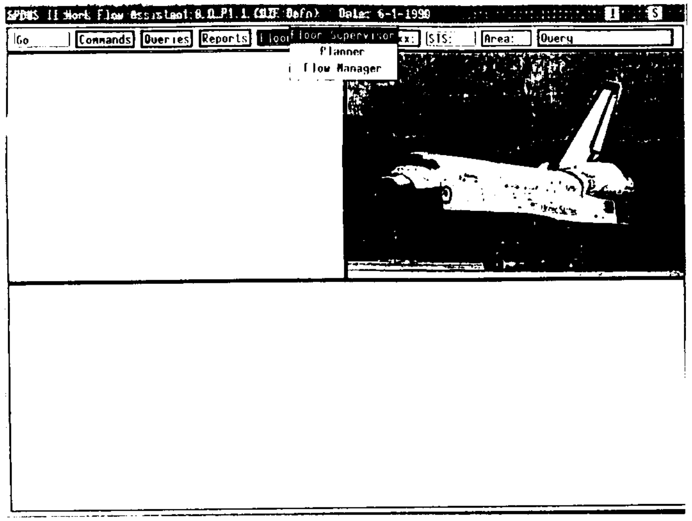
Each work center and user group have different needs for data content and form for the user interface as well as queries and reports. "One size fits all" is not appropriate in this context.

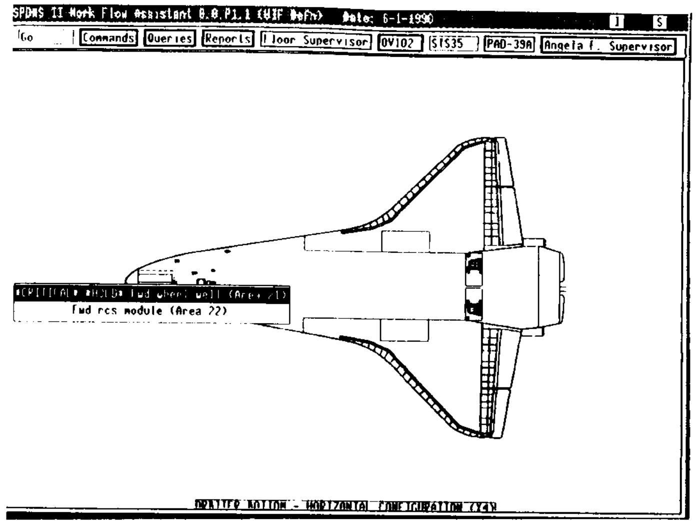
The WFA strategy is two fold. First, provide a core capability that is independent of specific work center or user group needs and provide technology for user customization. Secondly, turn over the customization responsibility and task to user groups.

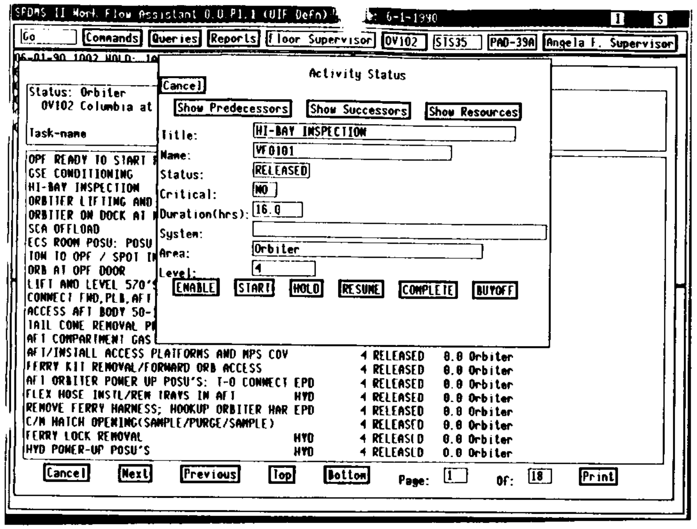
Since the SIIF is an SQL database, interactive query and reports are readily generated via the provided vendor SQL COTS tools. WFA does not provide these reports and queries but rather, ensures that all scheduling and status information is accessible through them.

The user interface for the assistants alternately presents information to the users as lists, drawings and various scheduling charts, all user selectable. The user can define as many drawings as needed to tailor the interface to the work area itself. Work center status can then be directly depicted on these drawings. This capability is presently being demonstrated to users and has been well received. Figure 2 illustrates various forms of the user interface for the assistants.

Advisory system displays are also fully defined by the users.




SPDMS II Work Flow Assistant 0.0.P1.1 (UIF Defn) Date: 6-1-1990
Go
Commands
Queries
Reports
Floor Supervisor
OV102
STS35
PAD-39A
Angela F. Supervisor
CRITICAL #H315# Fwd wheel well (Area 71)
Fwd rcs module (Area 72)
ORBITER BOTTOM - HORIZONTAL CONFIGURATION (X4)


SPDMS II Work Flow Assistant 0.0.P1.1 (UIF Defn) 6-1-1990

Go | Commands | Queries | Reports | Floor Supervisor | OV102 | STS35 | PAD-39A | Angela F. Supervisor

06-01-90 1002 HOLD

Status: Orbiter
OV102 Columbia at

Task-name

Activity Status

Cancel

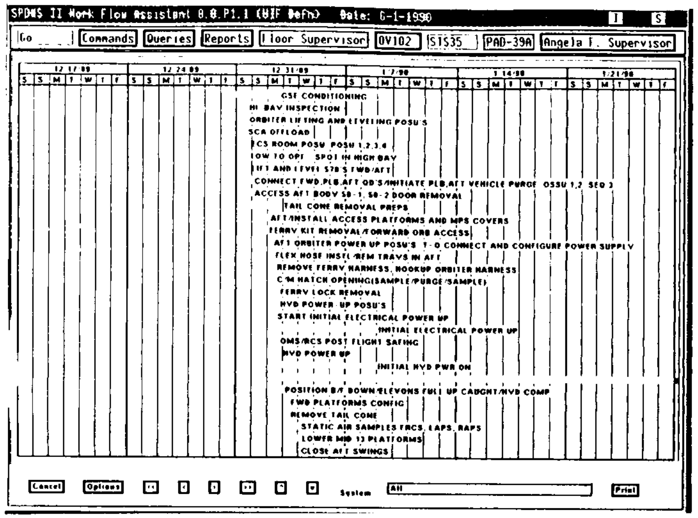
Show Predecessors | Show Successors | Show Resources

Title: HI-BAY INSPECTION
Name: VF0101
Status: RELEASED
Critical: NO
Duration(hrs): 16.0
System:
Area: Orbiter
Level: 4

ENABLE | START | HOLD | RESUME | COMPLETE | BUYOFF

```
OPF READY TO START
GSE CONDITIONING
HI-BAY INSPECTION
ORBITER LIFTING AND
ORBITER ON DOCK AT
SCA OFFLOAD
ECS ROOM POSU: POSU
TOW TO OPF / SPOT IN
ORB AT OPF DOOR
LIFT AND LEVEL 570'S
CONNECT FWD,PLB,AFT
ACCESS AFT BODY 50-
TAIL CONE REMOVAL PR
AFT COMPARTMENT GAS
AFT/INSTALL ACCESS PLATFORMS AND MPS COV          4 RELEASED   0.0 Orbiter
FERRY KIT REMOVAL/FORWARD ORB ACCESS             4 RELEASED   0.0 Orbiter
AFT ORBITER POWER UP POSU'S: T-0 CONNECT EPD     4 RELEASED   0.0 Orbiter
FLEX HOSE INSTL/REM TRAYS IN AFT             HYD 4 RELEASED   0.0 Orbiter
REMOVE FERRY HARNESS; HOOKUP ORBITER HAR EPD     4 RELEASED   0.0 Orbiter
C/M HATCH OPENING(SAMPLE/PURGE/SAMPLE)           4 RELEASED   0.0 Orbiter
FERRY LOCK REMOVAL                           HYD 4 RELEASED   0.0 Orbiter
HYD POWER-UP POSU'S                          HYD 4 RELEASED   0.0 Orbiter
```

Cancel | Next | Previous | Top | Bottom | Page: 1 of: 18 | Print


SPDMS II Work Flow Assistant 0.0.P1.1 (UIF Defn) Date: 6-1-1990

Go | Commands | Queries | Reports | Floor Supervisor | OV102 | STS35 | PAD-39A | Angela F. Supervisor

12/17/89 | 12/24/89 | 12/31/89 | 1/7/90 | 1/14/90 | 1/21/90

```
GSE CONDITIONING
HI BAY INSPECTION
ORBITER LIFTING AND LEVELING POSU'S
SCA OFFLOAD
ECS ROOM POSU POSU 1,2,3,4
TOW TO OPF SPOT IN HIGH BAY
LIFT AND LEVEL 570'S FWD/AFT
CONNECT FWD,PLB,AFT QD'S/INITIATE PLB,AFT VEHICLE PURGE OSSU 1,2 SEQ 3
ACCESS AFT BODY 50-1, 50-2 DOOR REMOVAL
TAIL CONE REMOVAL PREPS
AFT/INSTALL ACCESS PLATFORMS AND MPS COVERS
FERRY KIT REMOVAL/FORWARD ORB ACCESS
AFT ORBITER POWER UP POSU'S T-0 CONNECT AND CONFIGURE POWER SUPPLY
FLEX HOSE INSTL/REM TRAYS IN AFT
REMOVE FERRY HARNESS, HOOKUP ORBITER HARNESS
C/M HATCH OPENING(SAMPLE/PURGE/SAMPLE)
FERRY LOCK REMOVAL
HYD POWER UP POSU'S
START INITIAL ELECTRICAL POWER UP
INITIAL ELECTRICAL POWER UP
OMS/RCS POST FLIGHT SAFING
HYD POWER UP
INITIAL HYD PWR ON
POSITION B/F DOWN/ELEVONS FULL UP CAUGHT/HYD COMP
FWD PLATFORMS CONFIG
REMOVE TAIL CONE
STATIC AIR SAMPLES FRCS, LAPS, RAPS
LOWER MID 13 PLATFORMS
CLOSE AFT SWINGS
```

Cancel | Options | System: All | Print

Figure 2: Flexible User Interface

ORIGINAL PAGE IS OF POOR QUALITY

4. PROJECT BENEFITS

Program benefits range from aspects as diverse as technology insertion, and comparative validation/assessment of new products, to system maintenance. And as new products are introduced, they may be evaluated *in situ* rather than in a standalone mode.

The WFA will make information readily available to a large segment of the KSC user community, that has previously been assessable only via personal contact with numerous individuals with area specific knowledge.

Data sharing across tools, systems and organizational groups is a natural outgrowth.

Consumers of scheduling information and application developers as well should receive something new in tool selection...a choice.

Near real-time interactive decision making aids should improve productivity and permit more timely informed decisions

Once the proposed SIIF standard is in place, a synergistic effect may occur, and unforeseen latent benefits may accrue.

If successful, SIIF may encourage others within NASA or contractor organizations to undertake data format standardization efforts in other areas.

5. CONCLUSION

As reflected from user feedback of demonstrations of the first phase prototype, WFA is envisioned by users as a ground breaking project with much promise. However the challenge is also real. WFA is being therefore deployed in separate phases to mitigate that risk. Each major phase also contains a prototyping step to gain early user critique and permit project mid-course corrections. The success of this system and its extent are dependent upon factors which to a large measure require the persistent commitment and willingness of contractors, vendors and NASA to make WFA an actuality

ACKNOWLEDGEMENTS

The authors wish to thank Monte Zweben,et al of the NASA Ames Research Center for their contributing work involving the Ames Scheduler and Planner; and,Bob Gargen, et al of the Lockheed Artificial Intelligence Center, for their addition of Configuration Based Scheduling to the Ames project.

REFERENCES

Jamieson, J.R., Scarl, E.A., & Delaune, C.I., "A Knowledge Based Expert System for Propellant System Monitoring at the Kennedy Space Center", Proc. 22nd Space Congress, Cocoa Beach, FL,pp. 1-9.

Wilkinson, J., Tulley, J., "Automated Analysis of Shuttle Wiring Using SCAN", Proc. 25th Space Congress, Cocoa Beach, FL

Zweben, M., Deale, M., Eskey,M., "Anytime Rescheduling", Submitted to AAAI-90.

INTELLIGENT LAUNCH DECISION SUPPORT SYSTEM (ILDSS)

Beller
NASA
Hadaller and Ricci
Boeing

(Paper not provided at publication date)

N91-20681

EXODUS: Integrating Intelligent Systems for Launch Operations Support

Richard M. Adler and Bruce H. Cottman
Symbiotics, Inc.
875 Main Street
Cambridge, MA 02139

Abstract

NASA Kennedy Space Center (KSC) is developing knowledge-based systems to automate critical operations functions for the Space Shuttle fleet. Intelligent systems will monitor vehicle and ground support subsystems for anomalies, assist in isolating and managing faults, and plan and schedule Shuttle Operations activities. These applications are being developed independently of one another, using different representation schemes, reasoning and control models, and hardware platforms. KSC has recently initiated the EXODUS project to integrate these "standalone" applications into a unified, coordinated intelligent operations support system. EXODUS will be constructed using SOCIAL, a tool for developing distributed (intelligent) systems. This paper describes EXODUS, SOCIAL, and initial prototyping efforts using SOCIAL to integrate and coordinate selected EXODUS applications.

Section 1 Introduction

Over the past decade, NASA Kennedy Space Center (KSC) has developed knowledge-based systems to increase automation of operations support tasks for the Space Shuttle fleet. Major applications include: monitoring, fault isolation and management, and control of vehicle and ground support systems; operations support of the Shuttle Launch Processing System (LPS); and planning and scheduling of Shuttle and payload processing activities.

Initial prototypes have been tested successfully (off-line) in support of several Shuttle missions. KSC is currently extending and refining these systems for formal field testing and validation. The final deployment phase of development will integrate the knowledge-based applications, both with one another and with existing Shuttle operations support systems.

Integration will require solutions to many challenging problems. KSC's knowledge-based applications were developed independently of one another, using different representation schemes, reasoning and control models, software and hardware platforms. Knowledge and data bases are application-specific, as are external interfaces to users, LPS software, and LPS data channels. In addition, KSC's knowledge-based applications lack capabilities for modeling their peer systems and for communicating with one another across heterogeneous host platforms. This precludes working together cooperatively, for example, by sharing information and by coordinating complementary activities, to solve problems that the systems are incapable of resolving individually.

KSC has recently initiated the EXODUS project (Expert Systems for Operations Distributed Users) to investigate and address these difficult issues. A high-level integration architecture has been designed. The design incorporates a hierarchical distributed control model to coordinate cooperative efforts among KSC's intelligent operations support applications. In order to refine, test, and implement this design, KSC is funding Symbiotics, Inc. to develop SOCIAL, a generalized tool for integrating and coordinating distributed systems comprised of heterogeneous intelligent and conventional elements. Symbiotics is also developing proof-of-concept prototypes to validate SOCIAL and the proposed EXODUS architecture.

The remaining sections of this paper describe, in order: the EXODUS problem domain and system design; the SOCIAL development tool; and the demonstration prototypes that integrate and coordinate selected knowledge-based applications at KSC.

Section 2 EXODUS

2.1 Space Shuttle Ground Operations

Processing, testing, and launching of Shuttle vehicles takes place at facilities dispersed across the KSC complex, often using complex Ground Support Equipment. For example, Orbiters are mated to external tanks and solid rocket engines using cranes at the Vehicle Assembly Building. Propellant storage and loading systems are used to fuel Shuttle vehicles mounted on Mobile Launch Platforms at Launch Pads.

The Launch Processing System (LPS) supports all Shuttle preparation and test activities from arrival at KSC through to launch. The LPS provides the sole direct real-time interface between Shuttle engineers, Orbiter vehicles and payloads, and associated Ground Support Equipment [He87]. Four independent physical copies, called *Firing Rooms*, can support simultaneous processing of multiple Shuttle vehicles, LPS software development, and launch team training.

A Firing Room is an integrated network of computers, software, displays, controls, switches, data links and hardware interface devices (cf. Figure 1). The computers in a Firing Room are organized in a star network. The star's locus, called the Common Data Buffer, collects data, transfers data to LPS peripheral storage subsystems, and mediates computer–to–computer communications, which are concurrent and asynchronous. During peak (launch) conditions, a Firing Room handles thousands of commands and measurements per minute.

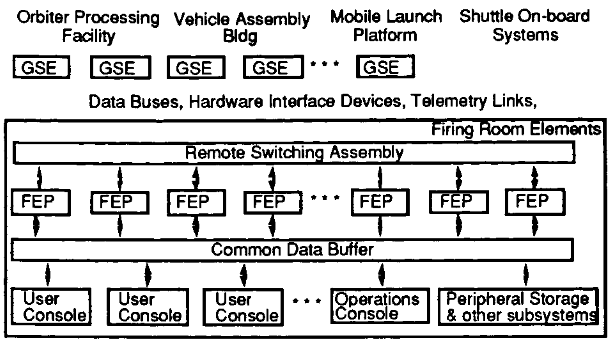


Figure .1: Architecture of an LPS Firing Room

Firing Room computers are configured to perform independent LPS functions through application software loads. Shuttle engineers use computers configured as Consoles to remotely monitor and control specific vehicle and Ground Support systems. Each such application Console communicates with an associated Front–End Processor computer that issues commands, polls sensors, and preprocesses sensor measurement data to detect significant changes and exceptional values. These computers are connected to data busses and telemetry channels that interface with Shuttles and Ground Support Equipment through switching assemblies in each Firing Room.

2.2 EXODUS Applications

EXODUS will integrate and coordinate knowledge-based applications that span KSC's major processing functions - Shuttle and LPS operations and planning and scheduling of such operations. Tasks in all three areas are labor– and expert–intensive. KSC's intelligent systems program will: increase automation of operations support tasks, alleviating labor requirements and costs; improve safety by standardizing (expert) task performance and increasing accessibility of data on problems and problem solutions; and preserve expertise that would otherwise be lost when veteran NASA engineers change jobs or retire. This section summarizes the primary KSC applications in the EXODUS framework.

The LPS Operations team ensures that the four Firing Rooms are available continuously, in appropriate error–free configurations to support Shuttle engineering test requirements such as Launch Countdown or Orbiter Power-up sequences. OPERA (for Operations Analyst) consists of an integrated collection of expert systems that automates some of these critical support functions [Ad89b].

OPERA's primary expert system monitors a Firing Room for anomalies and assists LPS Operations users in isolating and managing faults by recommending troubleshooting, recovery and/or workaround procedures. OPERA taps into and interprets a data stream comprised of error messages triggered by the LPS Operating System. Messages signal anomalous events such as improper register values or expiring process timers. OPERA also incorporates two secondary expert systems, which interface with and maintain data and knowledge bases that track open and recurring problems across all four Firing Rooms. They assist the primary expert by retrieving fault reports that provide relevant precedents to current problem symptoms.

The LPS Operations team replaces problem Firing Room computers with standby spares to restore on–line functionality to Shuttle engineering end–users. Suspect or faulty computers are then diagnosed and repaired off–line by an LPS Maintenance organization, which is developing a supporting Remote Monitoring and Maintenance Subsystem (RMMS). RMMS consists of custom hardware implants that capture memory dumps from failing Firing Room computers, and a tap to the Common Data Buffer for retrieving and storing dump data files. An associated Memory Dump Analyst provides an object–oriented interface for inspecting memory dumps and a shallow–knowledge expert system that automatically diagnoses a subset of computer faults.

KSC has developed a model–based tool called KATE (or Knowledge Based Autonomous Test Engineer) for building intelligent systems to automate monitoring, diagnosis, and control tasks for Shuttle Ground Support Equipment [Fu90]. These systems are comprised of electromechanical components including relays, pumps, blowers, ducts, heaters, and embedded sensors. KATE extends and generalizes on LES, an early model–based diagnostic system that supports the the Liquid Oxygen fuel loading system [Sc87].

KATE applications monitor Firing Room Console data while simultaneously running a behavioral model simulation for their target Ground Support Equipment system. Discrepancies between actual data and values predicted by the model trigger the model–based diagnostic module. Control capabilities can be used to test diagnostic hypotheses (via sensor requests) and to issue corrective commands. A KATE–based application called LOX (an extended reimplemented version of LES) is currently being validated in field tests. Another KATE system (ECS) has been developed to help maintain environmental controls for the Shuttle cargo bay when the vehicle is at a Launch Pad.

EXODUS will also integrate knowledge–based tools for planning and scheduling resources and activities for payload integration and Shuttle processing [Mu88,Zw89]. Further expert systems are being designed to assist LPS Operations in configuring Firing Room switching assemblies and to automate Shuttle engineering activities at application Console stations.

2.3 EXODUS Architecture

LPS Firing Room (ModComp-II) computers were built in the early 1970s. Their limited memory capacity is largely occupied by LPS Operating System and Shuttle user application software. Accordingly, KSC's knowledge–based systems have been implemented on other platforms, including Sun Workstations, Texas Instruments Explorer Lisp Machines, and PCs.

The proposed EXODUS architecture (cf. Figure 2) will use an Ethernet local area network for physically connecting intelligent application hosts. Intelligent systems will access LPS Firing Room data via an interface between the Common Data Buffer and a data concentrator. This interface currently extracts memory dump data for RMMS and Operating System error messages for OPERA. Extensions to support data and

ORIGINAL PAGE IS
OF POOR QUALITY

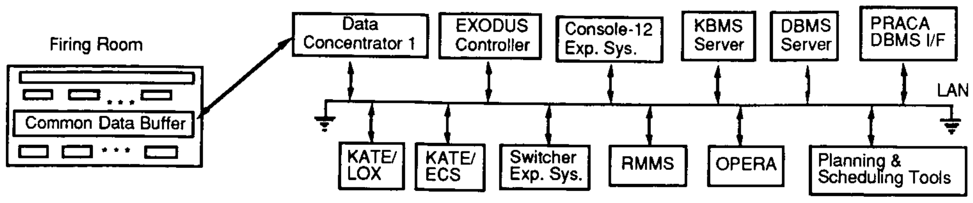


Figure .2: EXODUS Architecture

control interfaces for KATE applications are being designed. A centralized interface design is necessary for two reasons: (a) the limited number of free ports into Common Data Buffers; and (b) the major testing effort is required to validate and verify new LPS interfaces with respect to NASA's stringent safety requirements.

The proposed integration design for EXODUS adopts a server–based architecture: critical data and knowledge bases in EXODUS applications will be redistributed to server nodes comprised of dedicated data and knowledge base management systems running on high performance, large memory capacity hardware platforms. This design approach promotes sharing of symbolic models of common utility across applications: Shuttle and LPS system structures, behaviors, and bodies of operational expertise. Maintenance, access control, and commonality of interfaces will also be facilitated.

Redistributing large data and knowledge bases to server platforms will also reduce memory and performance burdens from EXODUS applications on their hosts. This will become critical since plans call for porting EXODUS applications over to the new Console computers being procured for a modernization of Firing Rooms in the mid–1990s.

The critical requirements for the proposed EXODUS integration architecture are: (a) non–intrusive communication capabilities for moving data and commands among heterogeneous applications and information resources; and (b) intelligent distributed control models to coordinate the activities of EXODUS applications. The following sections describe development efforts for these enabling technologies.

Section 3 The SOCIAL Development Tool

Obstacles to integrating "standalone" intelligent systems are not unique to KSC or to operations support. Analogous difficulties arise in other domains including: battle management; decision support; manufacturing process control; air traffic control; concurrent engineering environments; power generation plants; and power transmission and communication networks.

These domains encompass multiple problems of varying complexity, whose solutions may be independent or only weakly dependent upon one another. Different problem–solving architectures are appropriate for disparate tasks. Complex computer systems already exist for storing data and executing conventional programs that automate routine activities (e.g., for sensor and equipment control, instrumentation, event trapping, and bounded scheduling tasks). Software and hardware platforms are typically heterogeneous across intelligent and conventional applications. Finally, a priori design of comprehensive integration strategies was generally infeasible in the technology development or transfer environments where intelligent systems currently being deployed were initiated.

SOCIAL is a generalized tool that is being built for developing distributed systems and for integrating existing systems "after the fact" [Ad89a,Ad90]. SOCIAL will provide the following broad functional capabilities and attributes:

- a high–level, modular distributed communications capability for passing information between applications based on heterogeneous languages, platforms, networks, and network protocols. This subsystem is already available as a standalone commercial product called *MetaCourier*;
- minimally intrusive data and control interfaces to new and existing systems, both conventional and intelligent, including data feeds and applications developed using commercial AI shells and relational database management systems (RDBMSs);
- portability across heterogeneous software and hardware platforms;
- predefined intelligent control models to coordinate cooperative problem–solving activities of distributed (knowledge based) applications with heterogeneous internal control and communication architectures;
- tools for customizing and extending existing control models and interfaces.

SOCIAL's architecture is based on a layered library of object–oriented building blocks. The highest level objects are called *Agents*. Distributed systems are constructed by instantiating suitable Agent types, embedding application elements in these instances, and connecting the resulting Agents together. Agent instances provide generic distributed services to their embedded application elements. These services, implemented via lower–level object–oriented building blocks, include distributed communication, data and knowledge access, and control (e.g., process coordination, concurrency and reliability management).

Application elements access the distributed services of their embedding Agents through a high–level *Message–based* interface. For example, an application communicates with another via messages of the form (Tell :agent X :system Y message-contents). For each application Agent, the developer must define the expected form of incoming messages (i.e. an argument list), along with three procedural methods that specify: how to parse and process messages; test predicates for determining completion (i.e., in case the Agent dispatches messages to one or more other Agents for intermediate processing); and what results the embedded receiving application is to return. Auxiliary methods can be defined to simplify the organization of these primary Agent methods.

Message protocols determine the kind of communication behavior required for Agent interactions. The "Tell" protocol signals asynchronous behavior whereas "Tell-and-Block" indicates synchronous, "wait-and-see" behavior: an Agent that sends a Tell message can go on to perform other tasks pending returning information, whereas a Tell-and-Block message implements a function call and return control model.

All distributed control and information access behaviors are defined in terms of MetaCourier's message-based communication services, the substrate layer of the SOCIAL architecture. Distributed control is achieved through Agents autonomously invoking other Agents. For example, concurrency is accomplished by asynchronous message-passing to invoke multiple Agents more or less simultaneously. Similarly, parallelism amounts to dispatching subtasks (single or multiple instruction with multiple data) to a set of server Agents with a *broadcast* protocol of batched Tells. Non-intrusive access to, and integration of, passive data resources and existing standalone applications is accomplished through "wrapper" Agents that define suitable external command and data interfaces.

SOCIAL's message-based interfaces enforce a clean partitioning between application-specific functionality and predefined services such as distributed communications. To ensure portability, SOCIAL further isolates Agent dependencies on processing platforms, networks, and software environments (e.g. cpu, operating system, network type and host address, language compiler and editor), in separate (shared) "Host" and "Environment" objects. SOCIAL's MetaCourier subsystem uses message protocols and Host and Environment objects associated with the sending and receiving Agents to determine how to transmit messages across heterogeneous hardware and software platforms transparently. By separating and concealing the mechanical complexities of distributed processing, SOCIAL frees developers to concentrate on the architecture and behavior of their distributed applications. The first version of SOCIAL is scheduled to be completed at the end of 1990.

Section 4 EXODUS Prototypes

4.1 Distributed Data Transfer

The Data Concentrator is a critical component in the EXODUS architecture. It must concentrate, classify, and route real-time data to the intelligent subsystems responsible for monitoring Ground Support Equipment and Firing Rooms and isolating faults. A proof-of-concept simulation of these data transfer functions was constructed using SOCIAL. Figure 3 depicts the Firing Room data sources, EXODUS knowledge-based systems, and and hardware and software platforms for those systems. Network connections consist of Ethernet media and TCP/IP protocols.

The client/server Remote Procedural Call (RPC) model is the de facto communications standard today. This model is inherently synchronous, asymmetric, and pairwise: active clients request and block for services from reactive servers and a given client can only interact with a single type of server. Synchronous processing is unsuitable for the high volume data transfers required by EXODUS. The client-server model also forces the (active) data concentrator to be modeled as a router that sorts and feeds data to a set of client processes that use "requests" to transmit the data to (passive) EXODUS "server" applications. In contrast, SOCIAL's MetaCourier layer provides an asynchronous, symmetric, and peer-to-peer model. A single Agent can act as a client or a server or operate in both roles, and a "client" Agent can interact with multiple "server" Agents. In addition, behaviors can be inherited and/or specialized across Agent types.

The EXODUS simulation defines a single Data Concentrator Agent and a class of Data Injector Agents that are co-resident with the various intelligent Operations Support applications. The Data Concentrator receives and preprocesses Firing Room data. The concentrator agent then classifies and encapsulates the resulting data in MetaCourier messages, which are dispatched directly and asynchronously to relevant Injector Agents. Injectors inherit the structure and functionality of the Injector Agent class, specialized by a single dispatch method for injecting the data to the input interface for a particular application. The OPERA Data Injector, shown below, inserts data into a First-In-First-Out input buffer for CCMS Operating System messages. The RMMS Memory Dump Analyst Injector simply notifies users that new computer memory dumps are available for inspection.

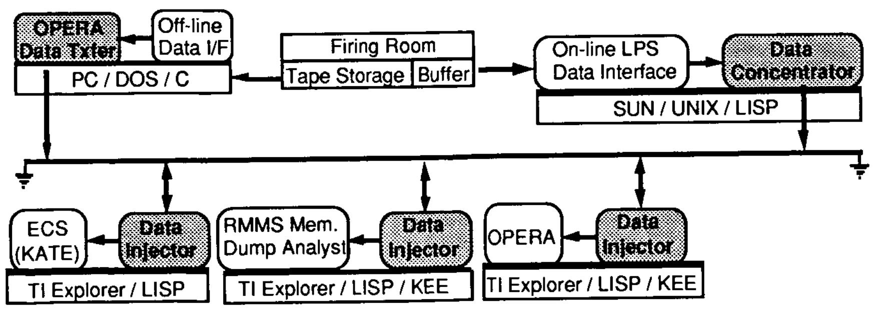


Figure .3: SOCIAL Exodus Data Transfer Simulation

ORIGINAL PAGE IS
OF POOR QUALITY

```
(defagent OPERA-DATA-INJECTOR
  :sys *opera-host*    ;;;OPERA host (vble)
  :environ 'exodus
  :args ($datum)       ;;;msg structure
  :lifetime :image
  :type (data-injector)  ;;; Agent class
  :documentation
  "This Agent inserts LPS Operating System
   error messages into the FIFO queue that
   serves as the OPERA LPS Data Interface"
  :in-filter ;;; inherited method/behavior
             ;;; to process incoming msg
 (sendx :self :dispatch-datum $datum)
  :methods
  ;;;OPERA-specific injector data interface
  ((:dispatch-datum ($data)
     (unless (string= $data "")
        (eval '(kee::add.value
           'kee::opera-controller
           'kee::opera-ccms-data-interface
           ,$data))))))
```

4.2 Distributing OPERA's Expert Systems

The capability to distribute a complex intelligent application across multiple platforms is critical for realizing EXODUS's resource server architecture. Physical distribution is clearly important for performance: time–intensive processes that search rule–bases or databases should be isolated, allocating dedicated computing resources to critical functions such as real–time data monitoring. Distribution of large knowledge bases also reduces memory loading. Because EXODUS encompasses existing applications, it must also be possible to *redistribute* application elements transparently and non–intrusively.

To demonstrate these capabilities, SOCIAL was used to physically distribute the OPERA system. OPERA is a logically distributed system that integrates and coordinates multiple expert systems that were originally developed as co–residents on a single platform. A control module coordinates the activities of OPERA's expert systems and manages all external interfaces. Expert systems request services from the Controller, which routes those tasks to appropriate servers. Expert systems post and retrieve task results from a shared memory "Bulletin–Board" on the Controller. OPERA's expert systems and Controller are integrated by embedding them within instances of a generic distributed blackboard structure, which provides standardized communications protocols [Ad89c].

Physical (re)distribution of OPERA elements was accomplished as follows (cf. Figure 4). The three primary blackboard protocols were altered to redirect communications as messages to MetaCourier Agents rather than as postings to other blackboards. Second, the OPERA Controller's service request routing table was extended to indicate a MetaCourier agent and host platform for each OPERA subsystem/blackboard. Third, MetaCourier Agents were written for each blackboard. The action of those Agents is simply to execute a protocol behavior that posts a message as an entry to the relevant structure on their associated blackboard. Finally, because distributed expert systems no longer have direct access to all OPERA information, additional messages were built into the protocols to ensure that information required to perform tasks was transmitted prior to task requests.

The redistribution experiment required roughly four days and one hundred lines of code. Extending the blackboard architecture using SOCIAL was quite simple. However, difficulties arose because the expert systems that were distributed depended on several common utility functions and data structures that were scattered across multiple source files and knowledge bases. The lesson drawn from this exercise is that these dependencies should be tracked as part of a standard development discipline for distributed systems. Such specifications would greatly simplify the (re)organization of system code and the identification of knowledge structures that need to be copied remotely.

4.3 Distributed Data and Knowledge Access

A third EXODUS requirement will be tools for developing non–intrusive interfaces to standalone applications and information resources. SOCIAL is addressing this need through "wrapper" Agent Types called Receptionists, which define bidirectional interfaces for passing control (i.e., commands), and data to the embedded resource or program.

Databases and application programs are often constructed using commercial development tools. The design of Receptionists for such systems can be simplified by abstracting the application-

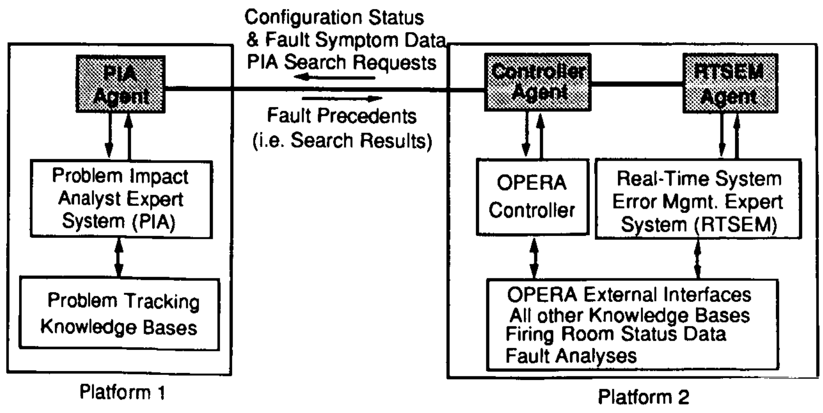
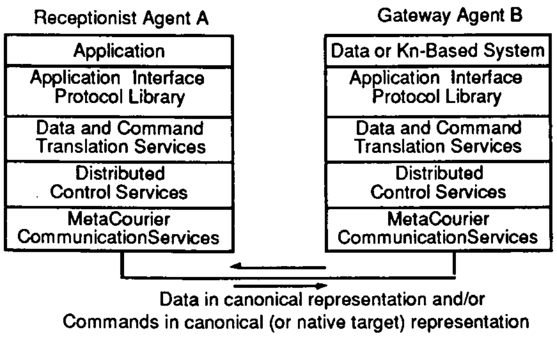


Figure .4: Using SOCIAL to Distribute OPERA

independent aspects of the control and data interface into a standardized, specialized Receptionist Agent type called a Gateway. Integrating an application element using a Gateway reduces to defining the application–specific aspects of the interface: the Gateway understands predefined query and command types which developers use to write specific queries or commands that name particular application objects and object attributes.

The basic operation of Gateways (or Receptionists) is depicted in Figure 5. An application's Agent sends a message to a Gateway Agent to access a protected resource or program. Depending on the situation, messages might contain data queries (i.e., read or write requests), or other commands to an application. Queries and commands may be expressed in a uniform, canonical language. An intelligent system might initiate queries or commands in its own development environment language through its Gateway to other SOCIAL Agents (including other Receptionists).

Gateways contain an interface library that maps canonical SOCIAL queries and commands into the language format of the relevant DBMS or shell environment, and vice versa. (Commands can be formulated in the target system's native language if desired, and will be passed through without alteration.) Gateways will also manage common exceptions (e.g., failed references or transactions), platform–specific data type conversions, and security features for restricting access to authorized Agents.



Receptionist Agent A
Application
Application Interface Protocol Library
Data and Command Translation Services
Distributed Control Services
MetaCourier CommunicationServices
Gateway Agent B
Data or Kn-Based System
Application Interface Protocol Library
Data and Command Translation Services
Distributed Control Services
MetaCourier CommunicationServices
Data in canonical representation and/or
Commands in canonical (or native target) representation

Figure .5: Integrating standalone systems using Gateways

An EXODUS simulation (cf. Figure 6) is currently being designed and implemented to demonstrate Gateway Agents for KEE, a LISP–based AI shell, CLIPS, NASA's C–based rule shell, and an Oracle relational DBMS. Briefly, OPERA will receive LPS error messages that indicate a failure in a Firing Room computer. OPERA will then request a reconfiguration action from the expert system for the Firing Room Switching Assembly. OPERA will then update its model of the Firing

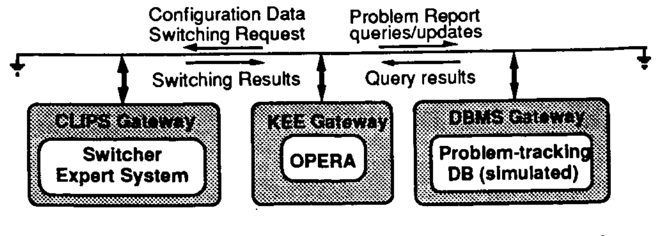


Figure .6: Distributed Data/Knowledge Access for EXODUS

Room based on the Switcher expert system and formulate error report entries to the Problem–Tracking Database.

4.4 Distributed Control

Aside from a robust communications substrate to provide the basic integration framework, the most important functional requirement for EXODUS is a capability to coordinate the activities of member applications. The proposed EXODUS architecture calls for a hierarchical distributed control model: a high-level Controller module will direct the intelligent applications described in Section 2 based on a global model of EXODUS subsystems, their associated KSC operations subdomains, and their relationships to one another.

SOCIAL will address this requirement through Agent types called Managers. A Manager Agent identifies all member (or subordinate) Agents by logical name and location, and also defines a distributed control model for organizing member Agents to work together cooperatively. It may also define specialized communication protocols for its members (e.g., one–to–many broadcast), and manage communication between member and outside Agents. Managers often provide a shared memory store of current problem–solving data for its members. Finally, Manager Agents may themselves be members of more complex organizations, subordinate to other Manager Agents.

The first Manager Agent type to be built for SOCIAL will be a reimplementation of OPERA's hierarchical distributed blackboard model (HDB) [Ad89c]. The HDB incorporates a routing table of member Agents describing their services and locations. The HDB also contains a centralized Bulletin–Board for expert systems to post service requests and post and retrieve request responses. The HDB control model routes all posted requests to suitable servers and orders and controls the activations of member expert system Agents. Member Agents can only communicate with one another indirectly, through the HDB Manager, using a common set of utility protocols for posting tasks to the HDB Manager's Agenda and posting results or checking for results on the HDB Manager's Bulletin–Board.

An EXODUS prototype is being planned that will utilize a Controller based on SOCIAL's HDB Manager Agent (cf. Figure 7). This Agent will coordinate KSC's intelligent systems for Shuttle and LPS Operations support to collectively solve a fault isolation problem that no single system could resolve individually. A test scenario will be defined in terms of LPS Operating System messages, Ground Support Equipment data, and Firing Room CPU memory dumps. The test scenario will simulate a Firing Room problem that may be caused by one of several possible fault candidates.

The EXODUS Controller will initialize member Agents and the Data Concentrator interface to a Firing Room. OPERA will process LPS error messages and inform the Controller of possible Firing Room anomalies. Because Firing Rooms lack adequate built–in test capabilities, OPERA can isolate fault candidates but cannot test them to produce an actual diagnosis. The EXODUS Controller will invoke the RMMS Memory Dump analyst expert system to investigate the possibility of a problem Console computer and also check KATE/LOX Agent to investigate the possibility of a failure in the Liquid Oxygen Subsystem. It will then use the hypothesis test results to reduce the set of fault candidates and display the results to Operations users.

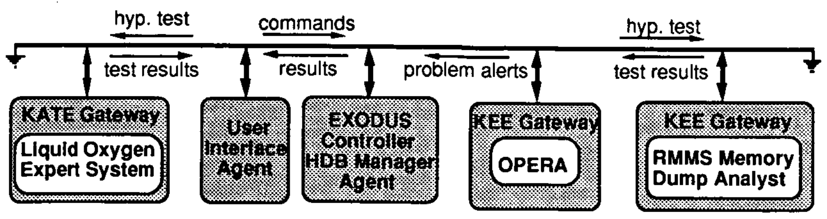


Figure .7: SOCIAL Exodus Distributed Cooperative Control

Summary

NASA Kennedy Space Center has initiated the EXODUS project to integrate and coordinate knowledge–based systems that are helping to automate Ground Operations activities in support of the Space Shuttle fleet. Individual applications were designed for "standalone" use with heterogeneous architectures, languages, and hardware platforms. Similar requirements exist for integrating conventional and knowledge–based systems in other Government and commercial domains. To minimize costly re–engineering, generalized integration tools must be developed that are non–intrusive, modular, and extensible.

KSC is using the SOCIAL development tool from Symbiotics, Inc. in the EXODUS effort. SOCIAL enforces a clear separation between application–specific functionality and standardized services for distributed communication, control, and data and knowledge access. Application elements invoke these services through high–level message–based interfaces to "wrapper" Agents, concealing the complexity and heterogeneity of the underlying distributed computing mechanisms and processing environments.

Proof–of–concept prototypes are described for validating the proposed EXODUS architecture using SOCIAL . These prototypes demonstrate SOCIAL's capability to support nonintrusive: distributed data transfer; physical distribution of a complex application comprised of previously co–resident expert systems and knowledge bases; cooperation of expert systems and data bases across multiple development tools; and hierarchical distributed coordination of standalone intelligent systems to solve difficult problems collectively.

Acknowledgments

Development of MetaCourier has been sponsored by the U.S. Army Signal Warfare Center, under Contract No. DAAB10-87-C-0053. Development of SOCIAL has been sponsored by NASA Kennedy Space Center under contract No. NAS10-11606. Astrid Heard initiated and oversees the EXODUS project at KSC. Rick Wood designed and implemented SOCIAL's distributed data and knowledge access capabilities. Pat Pinkowski and R. Bruce Hosken have provided valuable assistance in preparing EXODUS demonstrations at KSC.

Bibliography

[Ad89a] R.M. Adler, B. H. Cottman. "A Development Framework for Distributed Artificial Intelligence." *Proceedings Fifth Conference on AI Applications, Computer Society of the IEEE.* Miami, FL, March 6-10, 1989.

[Ad89b] R.M. Adler, A. Heard, and R. B. Hosken. "OPERA - An Expert Operations Analyst for A Distributed Computer Network." *Proceedings Annual AI Systems in Government Conference, Computer Society of the IEEE.* Washington, D.C., March 27-31, 1989.

[Ad89c] R.M. Adler. "A Distributed Blackboard Architecture for Integrating Loosely-Coupled Knowledge-Based Systems." *Intelligent Systems Review.* 1, 4, Summer, 1989, Association for Intelligent Systems Technology, E. Syracuse, NY.

[Ad90] R.M. Adler, B. H. Cottman. "A Development Framework for AI Based Distributed Operations Support Systems." *Proceedings Fifth Conference on AI for Space Applications.* Huntsville, AL, May 21-23, 1990.

[Fu90] S. Fulton and C. Pepe. "An Introduction to Model–Based Reasoning." *AI Expert.* 5, 1, January, 1990.

[He87] A.E. Heard. "The Launch Processing System with a Future Look to OPERA." *Acta Astronautica.* IAF-87-215, 1987.

[Mu88] A.M. Mulvehill. "A User Interface for a Knowledge–Based Planning and Scheduling System." *IEEE Transactions on Systems, Man, and Cybernetics.* SMC-18, 4, July/August 1988.

[Sc87] E.A. Scarl, J.R. Jameson, C.I. DeLaune."Diagnosis and Sensor Validation Through Knowledge of Structure and Function." *IEEE Transactions on Systems, Man, and Cybernetics.*SMC-17, 3, May/June 1987.

[Zw89] M. Zweben and M. Eskey. "Constraint Satisfaction with Delayed Evaluation." *Proceedings, 11th IJCAI.* Detroit, MI, Aug 20-25, 1989.

N91-20682

AN INTELLIGENT ADVISORY SYSTEM FOR PRE-LAUNCH PROCESSING

Peter A. Engrand, NASA
Tami Mitchell, NASA
Attn: TV-GDS-23
John F. Kennedy Space Center
Kennedy Space Center, FL 32899

Abstract:

In order to process Space Shuttle vehicles for launch, the various Shuttle systems are subjected to various test and checkout procedures prior to launch. The system of interest in this paper is the Shuttle Data Processing System (DPS) and, in particular, the DPS Multi-Function CRT Display System (MCDS). Due to the complexity of the Shuttle as a whole and DPS in particular, the system may at times behave in an unpredictable yet benign manner in respect to normal operations. Therefore, it is difficult for even experienced systems engineers to determine whether an annunciated error is truly a failure or a benign anomaly. An automated, prototype diagnostic tool is to be described in order to provide a solution to the labor intensive and time consuming diagnostic techniques currently used. The MCDS Diagnostic Tool (MDT) will be capable of monitoring the MCDS system real time, recognizing and analyzing failures giving the user a probable cause of the failure. The MDT is considered to be a pioneering diagnostic system for all DPS subsystems diagnostics.

The primary goal at the Kennedy Space Center is to prepare the space Shuttle system, both the Shuttle and its payload, for launch into low earth orbit. In order for this goal to be accomplished in an efficient yet safe manner, all Shuttle sub-systems are subjected to various test and checkout procedures prior to launch. A majority of these procedures are carried out by systems engineers (via ground software) from firing room resident computer consoles. This network of computer consoles which constitutes a main component of the ground Launch Processing System (LPS), is connected to the Shuttle via a Launch Data Bus (LDB). In this manner, systems engineers are able to monitor and control vehicle subsystems whether the Shuttle is residing in the orbiter processing facility, vehicle assembly building or pads.

The Shuttle system of interest in this paper is the Shuttle's Data Processing System (DPS). The DPS is composed of (1) General Purpose Computers (GPC) (2) Multi-Function CRT Display System (MCDS) (3) Mass Memory Units (MMU), and (4) Multiplexer/Demultiplexer (MDM) and related software. As is currently done, checkout and configuration of DPS subsystems are done by a DPS systems engineer initiating and controlling a set of ground software programs. Additional system configurations are done by manual switch settings inside the cockpit via voice instruction to a space craft operator.

In order to ensure the correct functioning of Shuttle systems, some level of automatic error detection has been incorporated into all Shuttle systems. For the DPS system, error detection equipment has been incorporated into all its subsystems. This error detection equipment is typically manifested as electronic circuitry composed of hardware registers where the bits of a particular register corresponds to particular errors (i.e. power transient detected). Additional errors are annunciated using both visual cues (i.e. mechanical flags and lights) and auditory cues (i.e. alarms and tones). This error detection equipment provides the system engineer with a real time awareness that a failure has occurred and allows him or her to properly safe the system in a timely manner. While this error detecting equipment makes the engineer aware of a subsystem failure, it does not (in most cases) give a cause of the failure. Due to the complexity of the Shuttle, both in terms of hardware and software, errors will frequently arise during normal operations which are of an anomalous but harmless nature, but again due to system complexity, it is up to the systems engineer level of experience to differentiate the harmful from the benign errors. Frequently, an inexperienced (and even an experienced engineer) will encounter a fundamentally benign failure yet diagnose it as a harmful one. In the interim, much paperwork is generated and a possible temporary interruption in launch processing may be experienced.

It is at this point that a brief description of the current diagnostic methods should be discussed. As was stated earlier, the error detection equipment alerts the responsible system engineer that an anomaly has occurred, but not what has caused it. In order to ascertain the cause of an anomaly, the responsible system engineer(s) must apprehend what the overall system environment was when the error occurred, and in order to do this, the engineer must rely on telemetry data. This telemetry data takes on two forms for DPS subsystems. The first is what is called dump data. Two of the DPS subsystems, the GPC's and the MCDS have resident stored memory capacity (GPC of 104K and the MCDS of 8K). When an error occurs in either one of these two subsystems, the malfunctioning component is isolated and the stored memory is then transmitted down the LDB and placed on magnetic tape and paper printout. This memory content of the anomalous subsystem provides the engineer with an image or state of operation of the subsystem during the time at which the error occurred. This information is then combined with the second type of telemetry which is called downlist data. Downlist data is simply the encoded values, states and times of a large number of discrete and analog Shuttle parameters. It is from this raw data which the engineer(s) must review manually, that a diagnosis of the problem is obtained (hopefully). Again, due to the complexity of the Shuttle, the amount of raw data is large (the MCDS alone has 8K of memory locations which must be reviewed manually) and, hence, is labor intensive and time consuming.

As stated in the abstract, we are describing an automated diagnostic system, the Multifunction CRT Display System Diagnostic Tool (MDT), which will aid in a more efficient processing of the DPS. Before going on to describe in more detail the functioning of the MDT, a brief description of the MCDS will be given. The MCDS is composed of three basic systems: (1) Keyboard, (2) Display Electronic Unit (DEU), (3) CRT. The heart of the MCDS is the DEU which is the information processor for data between the CRT, keyboard and GPC's and allows the astronauts to communicate to the GPC's and vice versa. The DEU is composed of various logical circuitry for CRT data display, keyboard data and processing of GPC commands and data. In addition, the DEU has a memory store of 8K in which to store data and commands for MCDS information processing, as well as, built-in test equipment (BITE) circuitry. This BITE circuitry is composed of three 16 bit status registers and two mechanical flags.

Now that a brief overview of the Shuttle processing and diagnostic environment has been described, a number of shortcomings have been mentioned in regard to the current nature of orbiter related diagnostics. These will now be stated more compactly:

(1) As it stands now, current diagnostic techniques used to arrive at problem resolutions are labor intensive and time consuming.

(2) Due to the complexity of the system (Shuttle, as a whole, and the DPS, in particular) being processed, the behavior of the system can, at times, behave in an unpredictable but benign manner with respect to normal operations. This makes it difficult for the systems engineers to know whether an annunciated error is truly serious or trivial.

Point 2 will have to be expanded upon, in order to make what follows in the rest of this paper consistent. The Shuttle has been in operation for almost 11 years, hence, there is also a commensurate 11 years worth of documented Shuttle behavior. The Shuttle behavior of most interest here is of the anomalous DPS kind, and this type of behavior has been, in most cases, thoroughly documented. This documentation will be described metaphorically as a triadic problem resolving knowledge base. The first part of this triad is known as a Problem Report (PR) database. This PR database is a paper system which is used to track the history and resolution (if one exists) of launch processing related anomalies. The second part of the triad is a "user's note" resource. The user note resource documents and explains DPS subsystem behavior which is anomalous but also benign with respect to the running of normal operations. The last and most important leg of this triad is the cerebral documentation which resides in the minds of our most experienced and astute engineers. It is from the inter-

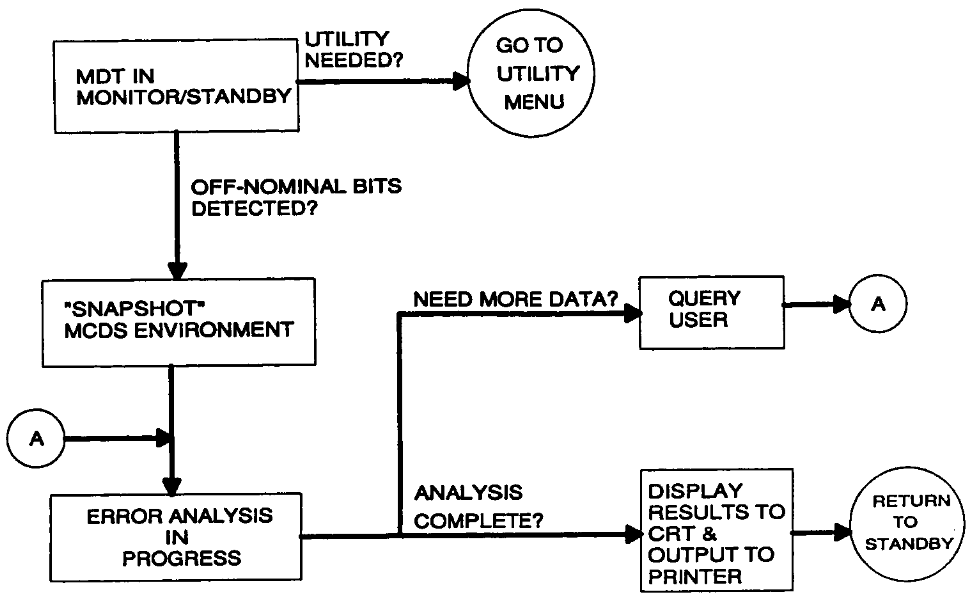

MDT IN MONITOR/STANDBY
UTILITY NEEDED?
GO TO UTILITY MENU
OFF-NOMINAL BITS DETECTED?
"SNAPSHOT" MCDS ENVIRONMENT
NEED MORE DATA?
QUERY USER
A
A
ERROR ANALYSIS IN PROGRESS
ANALYSIS COMPLETE?
DISPLAY RESULTS TO CRT & OUTPUT TO PRINTER
RETURN TO STANDBY

Figure 1.

action of this problem resolving knowledge base that solutions to our problems are derived.

The inherent drawbacks of this system are as stated in (1) above, and also in that our triadic system must rely on a component (i.e. that astute systems engineer) who may not be accessible for some reason or another when a problem arises. These two weaknesses in the system, by their very nature, allow themselves to be alleviated to some extent by automation. By incorporating a high-speed automated system, which has residing within it, both the paper history of problems and the diagnostic wisdom of our engineers (as best as that can be done), an integrative tool can be added to our diagnostic triad to help support the system as a whole. For our particular purposes (DPS), the MDT is a way of realizing an automated diagnostic system which can aid us in doing business with misbehaving avionics boxes. This system is to fulfill the following four goals:

(1) Monitoring of downlist data for MCDS anomalies

(2) Testing the downlist data for the presence of pre-defined error conditions

(3) Presentation to the user of probable cause of the failure

(4) Presentation of problem report and user note information corresponding to the failure detected.

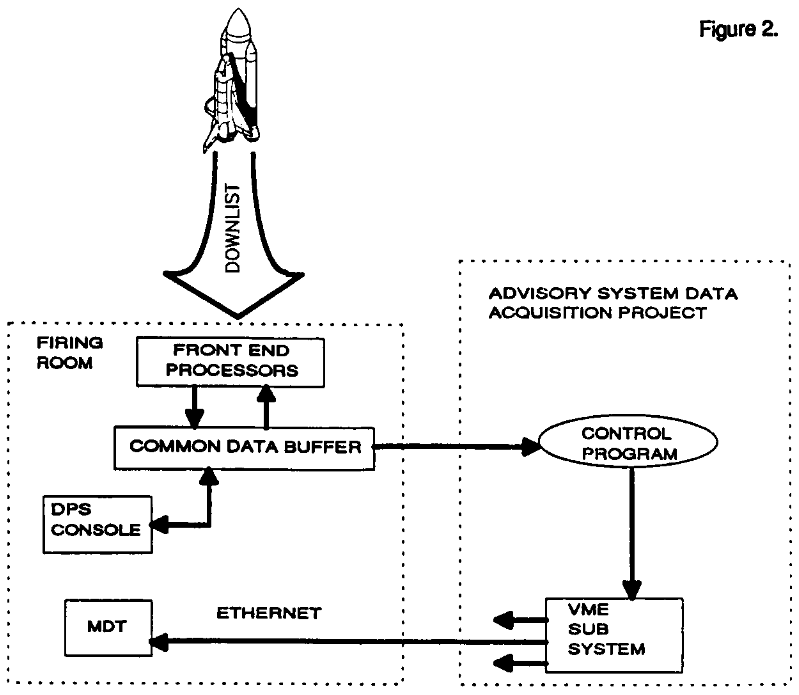
How the MDT proposes to attain these four goals is the topic of our next section.

The MDT is contracted to Rockwell International Corporation, Launch Support Services. The development team consists of Rockwell test personnel from the Avionics Software organization at KSC, Florida, and Artificial Intelligence personnel from the Expert Systems Applications organization at Downey, California. In addition, Abacus Programming Corporation personnel were added to augment the team.

The host computing system will be a SUN SPARCstation 1 Plus. This system was selected for its ability to perform multi-tasking with its UNIX based operating system. In addition, the SUN SPARCstation was selected to maintain compatibility between this advisory system and future firing room applications. The "C" Language Integrated Production System (CLIPS) was selected as the expert system shell to be utilized by the MDT. CLIPS is a forward chaining rule-based language that provides an inference engine and a language syntax that lends itself to interfacing with externally defined functions. As more and more firing room applications are automated, it is believed that a standard shell with increased capabilities will be available with the cost being shared among projects.

The conceptual system (Figure 1) will be powered on in the firing room at all times when the orbiter is powered on.

Figure 2.



The system will be in a "standby" mode of operation awaiting occurrence of off-nominal conditions. While in "standby" mode, test engineering personnel may pictorially view the current configuration and status of the MCDS system as it is on-board the orbiter. The MDT will dynamically update the System State Model display and monitor for errors with near real time data from a telemetry link to the firing room Common Data Buffer (CDBFR).

The process of acquiring the telemetry data (Figure 2) is a most challenging process, since the present firing room hardware is not compatible with the SUN. A method of acquiring telemetry data has been developed by the Advisory System Data Acquisition Project at KSC [3]. This telemetry data will be read from the Launch Processing System (LPS) common data buffer via a data control program. This program scans the buffer once a second and sends the data to a VME subsystem that blocks the incoming data stream into Ethernet frames and sends it out on an Ethernet line to various system users. The MDT communications process accepts the incoming data and places it in a data buffer residing on the SUN. This buffer is in shared memory and is accessible to the various applications resident on the SUN.

The incoming data will be monitored for off-nominal conditions. More explicitly, the three status registers for each of the four DEU's will be monitored for any abnormal bit pattern change. The monitoring of these 12 parameters can detect up to 90% of all MCDS failures. If an MCDS failure that is not triggered by status register changes occurs, the engineer will be able to utilize a manual mode of operations in which an analysis can be performed without being initiated by changes in telemetry data.

Once an anomalous condition is recognized, a "snapshot" of the MCDS environment is taken and saved for the system to begin the error analysis process. This snapshot process allows the system to complete the analysis process of a single error without possible loss of a secondary error. Once an error is detected, a set of pre-defined error conditions are checked. Each error condition requires the analysis of a unique set of data from the buffer and possibly the user. The user will be queried in situations where telemetry data is either unavailable or insufficient. In general, user requests will be data that may only be obtained by visual inspection of the MCDS (i.e. blank CRT's or tripped mechanical flags). The need for user supplied data will be kept to a minimum to enhance the automated nature of the system.

The system will evaluate and eliminate possible causes of the given failure condition. A hierarchy will exist among the pre-defined error conditions such that if an error condition of high probability is determined to be the cause, other unlikely conditions will not be checked. The possibility does exist for more than one probable cause of a failure to be displayed to the user. This could occur if telemetry data is temporarily unavailable or if the user did not supply the necessary data. As always, the systems engineer makes the ultimate decision concerning the most probable cause of the failure using data obtained from the MDT. The data used in the error analysis process will be saved to a file for later use in the re-creation of failure scenarios for either re-evaluation or training. The data necessary to perform the analysis process is being provided by Rockwell and NASA KSC employees. For each of the possible failure conditions currently recognized, a thorough review of historical data, both documented and undocumented, must be performed. This data is then organized into "rules" that will be encoded into the system by Rockwell, Downey.

All results of the error analysis will be displayed to the CRT and output to a printer. The results are the coordination of the system environment at the time of the error, probable causes of the failure, possible troubleshooting steps to be taken, and references to past problems and user notes pertaining to the failure condition. The coordination of this information is currently done manually by the systems engineers and is a time consuming process. All the information obtained from the results will be utilized to support closure of paperwork generated at KSC due to the MCDS failure.

The mode of operations described thus far constitutes the "Automatic" mode of operation. This mode will have the highest priority and will automatically be run as a foreground task upon receipt of an anomalous condition. This system also includes "Manual" and "Replay" modes that are available on an as-needed basis. The selection of either mode presents the user with sub menu's to access the MDT functions. The user will have the ability to perform "what-if" analysis on the MCDS in a test environment, replay already analyzed failures, review past PR and user note databases, and review the results of past failure analysis.

The MDT is viewed by the systems engineers at KSC as a highly desirable concept. By automating the processes performed manually at this time, MCDS failure recognition and resolution can be performed more rapidly and efficiently. This system will also provide invaluable training experience for systems engineers. The MDT system requirements and specifications, as defined, are such that the previously mentioned diagnostic triad representing an automated problem resolving knowledge base can be utilized.

The MDT is a 3 year project in which the first year prototype will concentrate on the development of a proof-of-concept prototype to demonstrate the four goals defined. The prototype will encompass the utilization of telemetry downlist data to perform diagnostics of the MCDS. The prototype to be delivered by October 1990 will perform automated analysis of between 5 and 10 errors. Its manual mode will provide the capability to search the PR and user note databases and to retrieve information about DEU status register bits. The second year will include the addition of the remaining known error scenarios, as well as, the addition of simulated DEU dump data. The dump data will enhance the diagnostic capabilities of the MDT. The final year will consist of integration with the Expert System for Operations Distributed Users System (EXODUS) [2] and the capture of near real time DEU dump data. This should complete the firing room implementation of the MDT.

The developers of the MDT view this concept as a pioneering diagnostic system for all DPS subsystems (GPC, MDM, MMU). A long term goal of the MDT project is that an eventual integration of all DPS diagnostics will be realized in a distributed diagnostic system. Such a distributed knowledge base concept for launch processing is already being investigated at KSC under the EXODUS program. It is hoped that the knowledge gained with the MDT and other Shuttle advisory systems currently being developed [1] will aid the EXODUS program and, in turn, help in the realization of a distributed DPS diagnostic system.

Acknowledgments

We would like to thank the following for their contributions:

Carol Lacey, Rockwell International, LSS, KSC

Anton Melichar, Rockwell International, LSS, KSC

Thomas Fowler, Abacus Programming Corporation

Warren Lackie, NASA Projects Office, KSC, and Advisory System Data Acquisition Project

References

[1] R. Adler, A. Heard, R. B. Hosken, "OPERA - An Expert Operations Analyst for a Distributed Computer System", AI Systems in Government Conference Proceedings, March 1989.

[2] A. Heard, "Distributed Knowledge-Based Systems for Shuttle Firing Rooms", NASA KSC Advanced Project Office, January 1990.

[3] W. Lackie, "Launch Countdown Trend Analysis, Launch Characteristic Analog Data LCAD A KSC Demonstration Project", NASA KSC Shuttle Engineering Flight Project Office, March 1989.

APPENDIX

Acronyms

BITE - Built In Test Equipment
CDBFR - Common Data Buffer
CLIPS - "C" Language Integrated Production System
CRT - Cathode Ray Tube
DEU - Display Electronics Unit
DPS - Data Processing System
EXODUS - Expert System for Operations Distributed Users System
GPC - General Purpose Computer
LDB - Launch Data Bus
LPS - Launch Processing System
MCDS - Multi-Function CRT Display System
MDM - Multiplexer Demultiplexer
MDT - MCDS Diagnostic Tool
MMU - Mass Memory Unit
PR - Problem Report

DIAGNOSTICS

N91-20683

INTELLIGENT MONITORING AND DIAGNOSIS SYSTEMS FOR THE SPACE STATION FREEDOM ECLSS

Brandon S. Dewberry
NASA/MSFC/EB42
Systems Software Branch
MSFC, AL 35812
(205) 544-4247
NASAMail: BDEWBERRY

James R. Carnes
Boeing AI Center
P.O. Box 240002, M/S JA-74
Huntsville, AL 35824-6402
(205) 461-2348
email: ray@huntsai.boeing.com

A B S T R A C T

This paper describes specific activities in NASA's Environmental Control and Life Support System (ECLSS) Advanced Automation Project designed to minimize the crew and ground manpower needed for operations. We will describe various analyses and the development of intelligent software for the initial and evolutionary Space Station Freedom (SSF) ECLSS. The paper describes: (1) intelligent monitoring and diagnostics applications under development for the ECLSS domain, (2) integration into the MSFC ECLSS hardware testbed, (3) an evolutionary path from the baseline ECLSS automation to the more advanced ECLSS automation processes.

The Environmental Control and Life Support System is a Space Station Freedom distributed system with inherent applicability to extensive automation primarily due to its comparatively long control system latencies. These allow longer contemplation times in which to form a more intelligent control strategy and to prevent and diagnose faults. The regenerative nature of the Space Station Freedom ECLSS will contribute closed loop complexities never before encountered in life support systems.

1. INTRODUCTION

The Environmental Control and Life Support System (ECLSS) aboard Space Station Freedom will sustain a safe shirt sleeve environment for its crew and payloads. Development has been divided into six functionally interconnected subsystems (Figure 1): Temperature and Humidity Control (THC), Waste Management (WM), Fire Detection and Suppression (FDS), Atmosphere Control and Supply (ACS), Water Recovery Management (WRM), and Air Revitalization (AR). The last two subsystems, WRM and AR, close air and water environmental loops to an extent never before attempted in space, and will require new technologies which are now undergoing extensive test and analysis.

1.1 ECLSS Background

Evaluation of the baselined and evolutionary ECLSS water recovery and air revitalization subsystems is continuing in NASA's Core Module Integration Facility (CMIF) and in several SSFP Work Package One development testbeds, all in Building 4755 at Marshall Space Flight Center (MSFC). These testbeds provide an enclosed environment in which regenerative ECLSS components are developed and tested for extended durations, while data is gathered and distributed to various analysis computers and personnel. Component and system tests are specifically designed to help engineers, biologists, and medical experts refine the technical specifications for the regenerative systems, WRM and AR.

The ECLSS is required to be as autonomous as possible to free crew for less mundane activities and to promote system growth. New components and procedures will be introduced to the system as if evolves. The baseline ECLSS is quite dynamic and will have a variety subsystems either functioning, or in a state of reserve, maintenance or repair.

Managing the operation of any one ECLSS subsystem is a formidable task taxing the current state of practice in software engineering. Although, as stated earlier, the ECLSS is a set of highly interactive subsystems, whose interaction has been isolated to a set of well-controlled water and gas buffers. However, these interactions, and the operations of the system as a whole, cannot necessarily be expressed in engineering terms given the atmospheric, chemical and biological processes defined across the various interfaces[5]. In the baseline system, crew members will be required to "tune" the system to achieve specific performance parameters dependent on two or more ECLSS subsystems.

As knowledge based systems are well-suited for controlled searches of large amounts of reconfigurable data, the use of knowledge-based system processes may provide enhanced capabilities to meet these needs within the baseline Space Station Freedom computing environment. The knowledge structures used by these systems may also serve to store important data for future reference and training.

1.2 Project Objectives

Preliminary study limited the scope of the domain to the potable water, hygiene water, and air revitalization problem analysis since they are functionally complex, yet still amenable to

C-5

knowledge-based solution. While a detailed ECLSS automation system evaluation revealed several viable applications within that domain, the potable water recovery fault detection, isolation and recovery (FDIR) functions were used in early prototyping. This limited the effort to a reasonable size, while providing a proof-of-concept and driving a detailed requirements derivation process. The early prototyping domain was selected based on: the abundance of knowledge, high level of visibility, and need to accelerate advanced functionality for advanced automation. The potable water system was prototyped by knowledge engineers working with ECLSS technical experts.

The current objectives for this project are to demonstrate fault detection, isolation and recovery capabilities at the subsystem level for the Potable Water, Hygiene Water, CO2 Reduction and CO2 Removal Processes, and ECLSS system level control, diagnostics, and trends. To accomplish these objectives we are integrating different advanced technologies such as knowledge acquisition, model-based reasoning, distributed computing to support software development and problem solution. We are leveraging our software development process with knowledge engineering tools from other NASA or Boeing projects including: Aquinas for knowledge acquisition, ART/Ada Automated Reasoning Tool shell for associational reasoning, KATE for model-based reasoning, and Erasmus for distributed blackboard operations. One of the strong goals for this project is to demonstrate and document a growth path for baseline software functions into intelligent systems. This paper provides background description of the Space Station ECLSS then focuses on the diagnosis methodology and implementation.

2. ECLSS DESIGN

Life Support Systems are required to provide the habitable environment for the crew and life sciences payloads. This environment includes water for drinking and washing, and atmospheric gasses. Previous life support systems have typically met these requirements by maintaining sufficient supplies of pressurized gasses and fluids, through closed loop options have been investigated[6].

2.1 Baseline Process Description

The Temperature and Humidity Control, Water Recovery Management, and Air Revitalization Subsystems aboard the Space Station combine to meet the water and air supply requirements as in Figure 1. These requirements are met by closing the air and water loops to an extent never before implemented in space. Even so, the control system is essentially open loop, a batch filtering process. Little chemical or microbial data is fed back into the control system for use in adjusting flexible processes for maximum efficiency.

The system is tested on the ground for sufficient cleaning and recycling set types and levels of fluids in the air and water, and is periodically verified on orbit using batch laboratory analysis procedures. This alone, the actual integration of these multiple interacting subsystems to specified requirements, will be a great achievement. Lessons learned in the on-orbit integration of these batch processing systems will be invaluable in determining micro-gravity interactions and recombinations of chemical and microbial constituents throughout the revitalization systems.

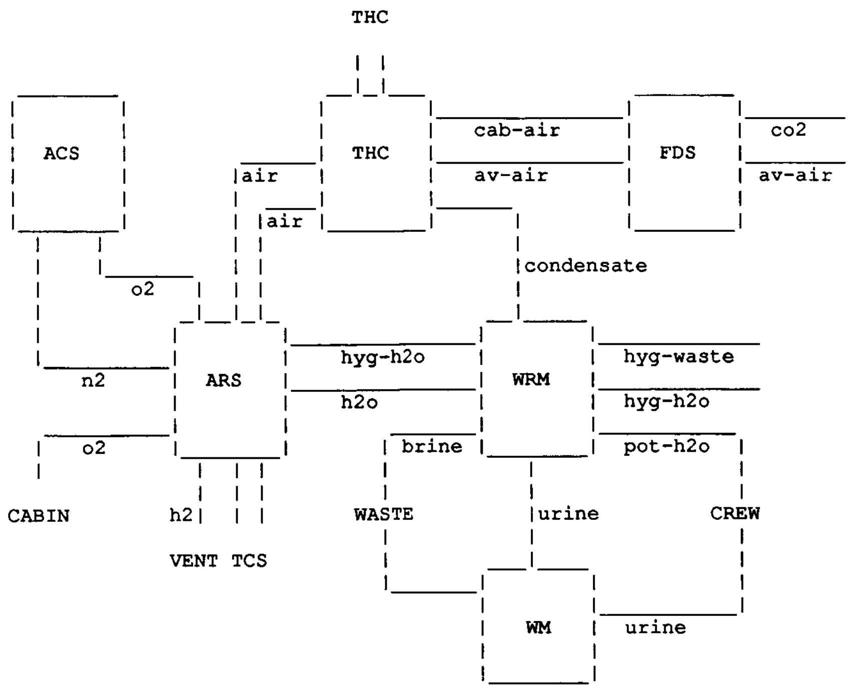


FIGURE 1 - ECLSS FUNCTIONALLY INTERCONNECTED SUBSYSTEMS

Two software processes which were determined prime candidates for automation are Real-time and Off-line Subsystem FDIR (Fault Detection, Isolation, and Recovery), and Component Performance and Trend Analysis. Both of these processes will contain parts initially in the ECLSS Ground Sustaining Engineering, with migration on-board when flight data management resources permit. An overview of the software architecture for the ECLSS can be found in reference [5].

2.2 Advanced Regenerative Life Support System

In general, future autonomous regenerative life support systems, including the evolutionary ECLSS, will be required to supply water and air, within specific chemical and microbial limits, for extended durations without crew or ground support adjustment. The control system and plant will be intelligent and robust enough to autonomously withstand unexpected crew and payload anomalies. These requirements will be achieved with a minimal set of instrumentation and processing assemblies.

These requirements may be met by augmenting the baseline ECLSS with various technologies. Software hooks and hardware scars in the baseline will be necessary to minimize the impact of integrating these technologies after Assembly Complete. Increased automation of the ECLSS is possible, but evolution to complete automation, defined as above but requiring some simple unit replacement occasionally, may not be feasible due to the degree of fundamental process adjustments and control strategies required. But the ECLSS can be used to dramatically increase the state-of-the-art in regenerative life support systems.

There are several advantages to beginning ECLSS automation with upgrades in the automatic fault isolation and recovery and health maintenance (failure prediction and prevention) processes. These processes are software oriented and theoretically, software is the most flexible part of the system and most amenable to upgrade.

Automatic fault isolation and recovery (FDIR) and health maintenance (failure prediction and prevention) processes require the implementation of emerging software technologies. These processes can be verified in the ground support environment and migrated to the flight ECLSS to increase the Station's flight autonomy. This approach to increasing ECLSS autonomy is described in [3] and [4] and is be the focus of the ECLSS Advanced Automation Project.

3. DIAGNOSIS APPROACH

We have divided the diagnosis problem into three layers: Reactionary, Heuristic and Comprehensive. Reactionary diagnoses are easily classified at low levels in the component structure and are usually within the scope of and implemented in the control logic (i.e., look-up tables) of a single component. They are often manifest as caution and warning statements in human interfaces. Specification of these so called "reactionary diagnoses" is already built-in to the baseline ECLSS design and will not be covered in this paper.

Heuristic diagnoses are characterized with some degree of accuracy (or confidence) by selected "rules-of-thumb". These rules usually associate specific component and system state information is a diagnostic observation (e.g., a component fault warning). This type of analysis is presented below in our work on associational diagnosis.

Comprehensive diagnoses are characterized, in our system, by component-oriented models of system structure and function. These models provide a stronger, perhaps more detailed definition of each system component; however, the major distinction of this diagnosis class is the causal propagation of structural and functional data through component networks to determine (and discriminate among) a set of diagnostic hypotheses. This type of analysis is presented below in our work on model-based diagnosis.

3.1 Associational Diagnosis

The associational diagnosis applications are designed to function quickly, constructing diagnostic observations about the functioning (or malfunctioning) system through standard forward- and backward-chaining mechanisms. Component-oriented rules used here are shallow and and are quite sensitive to system configuration/mode and component state changes. However, the ECLSS environment can be partitioned into a small of major operating configurations and system modes making feasible the use of this heuristic rule-based approach to diagnosis without a combinatorial explosion of rules. The ultimate goal for this module is to construct a diagnosis approach (that is compact in size, yet broad in scope) for integration with ECLSS flight software on-board the Space Station.

We are currently working on two approaches to associational diagnoses. The first approach is strictly heuristic mapping a set of abstracted system states into a set of possible component diagnoses with confidence levels. The mapping between system states and component diagnoses are acquired and managed with a knowledge acquisition workbench, Aquinas (from Boeing). Aquinas gathers the complex relationships between system traits and diagnostic solutions from one or more experts and stores it in a hierarchical network of repertory grids[11]. This knowledge can be examined and refined using tools that do clustering, similarity analysis, implication analysis, and consultation testing. These tools use techniques to analyze the information in the grids and suggest way to refine the knowledge base. After the diagnostic knowledge in an Aquinas grid is verified by the ECLSS engineers and ready for operational use, it is encoded into a set of ART/Ada (from Inference) facts and rules.

The second approach employs a component-oriented model-base written in G2 (from Gensym), a real-time expert system shell. G2 can be used to develop the same type of heuristic diagnosis model as described above, however, causal models can also be defined and used with the G2's built-in simulation engine to propagate functional properties through a network of components. When the simulation engine is run in parallel with the real-time system, the simulated property values in each component can be compared with the analogous observed values. If a discrepancy is noted diagnostic rules that reference the anomalous values are activated and diagnostic reasoning runs through matching and resolution mechanisms to produce forward- and backward-chaining effects. This approach provides an excellent architecture for system monitoring and a stronger (than the first approach) to diagnosis with an approach for focussing the inference engine on specific diagnosis rule, increasing the manageability of the rule set and decreasing the response time required for diagnostic analysis. However, a major weakness in this approach, as in the first approach, is that a specific diagnosis can be rendered only if it was preconceive and programmed into the rule set.

3.2 Model-Based Diagnosis

Component-oriented definitions are also used in the model-based approach but are in more detail and are quite robust in their reaction to system configuration/mode and component state changes. The near-term goal for this module is to construct a diagnosis approach to appraise the overall system health from a ground-based site integrated with ECLSS ground support software.

The model-based diagnosis application in this project is accomplished with NASA's KATE (Knowledge-based Autonomous Test Engineer) software. Models of the ECLSS subsystem processes are being constructed and refined using the KATE definition language. The KATE knowledge base use a frame representation to model the system processes. Specifically each component's interfaces, functions, measurement and command structure defined within the slots of selected frames. An example of a component definition in KATE's declarative-style definition language is shown in Figure 2.

```
(DEFRAME PUMP
  (NOMENCLATURE "a pump")
  (AKO ANALOG-OBJECT)
  (INSTANCES PUMP1)
  (INPUTS (IN1))
  (OUTPUTS (OUT1) (OUT2))
  (OUTPUT-FUNCTIONS
    (OUT1 (* IN1 PUMP-OUT-SELECT))
    (OUT2 (- IN1 (* IN1 PUMP-OUT-SELECT))))
  (PARAMETERS (PUMP-OUT-SELECT 0.5))
  (DELAY (OUT1 2) (OUT2 2))
  (TOLERANCE (OUT1 0.1) (OUT2 0.2))
  (UNITS "ml/min"))
```

FIGURE 2 - SAMPLE KATE OBJECT DEFINITION

The diagnosis algorithm in KATE scans a set of observed measurements comparing them to a set of simulated values obtained by propagating commands forward through the network of components models. Once some measurement has been noticed to be discrepant, the diagnoser is invoked to localize the fault to the extent possible. Faults are perturbations from a system's expected functionality. Diagnosis, in this case, is the search for one or more faults that can explain the system's observed behavior. The strength of component-oriented modeling lies in its ability to hypothesize faults from the information given by discrepant sensor readings[7].

A general analysis generates possible fault hypotheses for possible faulty objects, these fault hypotheses predict hypothesized sensor measurements, and those measurement hypotheses are tested against observed sensor readings. An agreement of fault hypothesis with observation lends support to (but does not prove) the hypothesis, while a contradiction would rule out that hypothesis. If all fault hypotheses for a particular faulty object are ruled out, then that object is no longer suspect.

Fortunately, one does not need to test all suspects (potentially faultly objects) against all related sensors. Search is anchored by the discrepant sensor or sensors. Only objects which are connected in controlling relationships to a discrepant sensor-object are considered as potential suspects. The diagnostic algorithm is discussed in more detail in [10]. An earlier, more structurally-oriented diagnostic algorithm is discussed in [9].

The greatest savings comes from using discrepant sensors to reduce or even eliminate the search for hypothetical faults by transmitting the information in their readings to the suspects. This means effectively inverting the dependency of sensor upon suspect which is known through the interface and function expressions. Such inversion generates all hypotheses consistent with discrepant sensor readings, and even these are eliminated where possible by contradiction with other sensors.

4. PROJECT ARCHITECTURE

The major goal of this project is to produce intelligent system software to monitor, control, and diagnosis the Space Station ECLSS. Five major components of this software system are under development in a distributed environment: console interface, model management, data acquisition, associational reasoning, and model-based reasoning. The applications for diagnostic reasoning have already been discussed in detail in the previous section. Figure 3 illustrates the relationships between the software subsystems in this project.

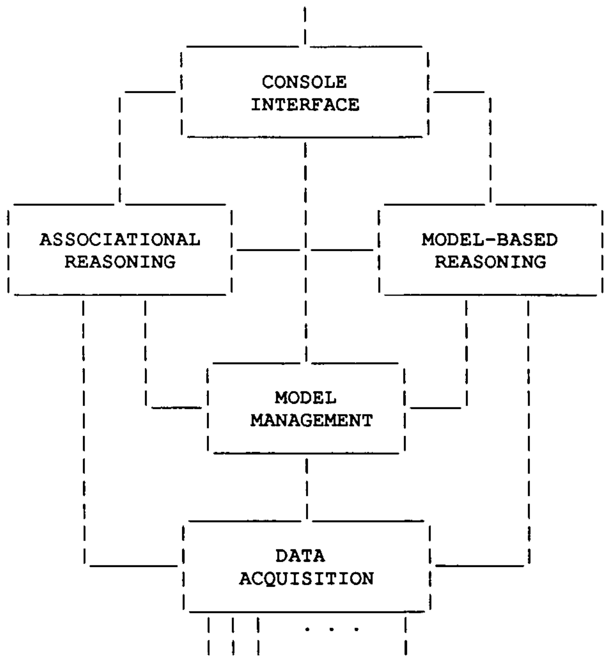


FIGURE 3 - ADVANCED ECLSS AUTOMATION ARCHITECTURE

4.1 Console Interface

The Console Interface module provides human computer interaction for monitor and control applications running in different modules. The look-and-feel of the interface conform to the Space Station Freedom Program Work Package 02 standard SY-45.1[2] and is being developed in TAE-plus, an X-window-oriented application tool for prototyping computer interfaces for both flight controls and ground consoles. The Console Interface receives information on current system state/status from sensor and actuator data (formatted by the Model Manager) and updates on monitoring and diagnostics applications for the Associational and Model Based Reasoners. Control commands for the domain system can be formed and issued through the Model Manager, while control of monitoring and diagnosis applications is fed to the appropriate reasoning module.

4.2 Model Manager

The Model Manager is a module to store and control access to the object knowledge base and run-time database. It provides a consistent definition of components/systems to the control and diagnostic algorithms running in the Associational and Model Based Reasoning modules. The Model Manager also manages the run-time collection of the ECLSS environment collecting observations of the sensors/actuators from the data acquisition module.

4.3 Data Acquisition

The Data Acquisition module for the system is provides binding to all sensors and actuators running in the hardware testbed. Data collection for the prototype software is accomplished through a modification to the existing SCATS (Systems and Components Automated Test System) data server used to supply data to the control panels test bed control room.

4.4 Diagnostic Modules

The independent diagnosis approaches described in the previous section are being integrated into the reasoning modules and coupled with the Model Manager. Structural and functional models of the ECLSS subsystem processes are used to diagnose and isolate failures. The model based approach to diagnosis is computationally intensive but performs autonomous, in-depth diagnosis of faults. The process control nature of the ECLSS allows the use of emerging model based reasoning tools in automating the system, while storing knowledge in component form[7]. The system also may be upgraded for automatic diagnosis of regeneration analysis with the future inclusion of chemical and microbial transfer equations.

We have analyzed and developed detailed models of two different processes within the WRM subsystem (Potable Water System and Hygiene Water System). KATE and G2 models have been constructed for the Hygiene Water process. The work with the process models focuses on integrating multiple-aspect models (i.e., structural, functional, thermal, etc.) as opposed to the explicit modeling of disfunction.

Associational failure models for the Potable Water process were developed using Aquinas. Current models establish relationships, in hierarchical grids, down to a level below the ORUs (Orbital Replacement Units). A functional architecture for the integration is depicted below.

4.5 Testbed Description

The following diagram (Figure 4) depicts the hardware and software configuration currently in use to support prototype development for the CMIF Testbed.

```
                                               prototype | testbed
                                               equipment   equipment
                                                     <---|--->
(system monitoring      (fault management
  and diagnosis)        performance testbed)             |
                                                                o testbed data
  o TAE-plus               o TAE-plus                    |        collection
  o Gensym/G2              o ART/Ada                              distribution
                                                         |
  ____________           _____________                         _____________
 |            |         |             |                        |             |
 |    SUN     |         |     IBM     |                  |     |    DEC      |
 |  Sparc-1   |         |  PC-80386   |                        |  VAX 6320   |
 |____________|         |_____________|                  |     |_____________|
       |                      |                                       |
       |                      |                          |            |
=========================================================================
ethernet       |                      |                  |              |
         ______|______          ______|______                  _______|______
        |             |        |             |           |     |             |
        | Symbolics   |        | Symbolics   |                 |  Subsystem  |
        |   3670      |        |   3650      |           |     |  Hardware   |
        |_____________|        |_____________|                 |_____________|
                                                         |
         o Gensym/G2            o KATE                       o potable water process
                                                         |   o hygiene water process
        (subsystem             (subsystem                    o co2 reduction process
         simulation             simulation               |   o co2 removal process
         and monitoring)        and diagnosis)
```

FIGURE 4 - EXAMPLE ARCHITECTURE FOR TESTBED INTEGRATION

5. CONCLUSION

The Environmental Control and Life Support System aboard the Space Station Freedom will be a step ahead in the implementation of regenerative life support systems. The interactions of its subsystems with each other and the crew will serve to greatly increase our knowledge in low gravity regenerations complexities. The Space Station can be used as a test bed for verification of chemical and microbial, variable gravity transfer models which will prove essential in long duration regenerative life support system engineering and autonomy analysis.

The fully automated regenerative life support system described cannot be built today. Quite a few steps must be taken, and research performed in order to develop systems which can autonomously remain stable for long durations. A first step is to build and deploy the Freedom Station. The actual hands-on knowledge generated from ground and flight test will allow incremental builds upon the ECLSS toward automation and long term stability. Another step is the inclusion of the Life Sciences medical technology in Life Support engineering. Life support systems which use regenerative techniques to meet their supply requirements will have to actively worry about and control microbial recombination, and insure

To support this work future work in Aquinas will include automatic generation of ART/Ada rules from grid structures. Future work in KATE will include simultaneous equation solving and constraint suspension to provide more flexibility in modeling physical systems and more discriminatory power in diagnosis.

6. REFERENCES

[1] Carnes, J. R., "Model-Based Diagnostics," in the Intelligent Sensory Processing Workshop, IEEE International Conference on Robotics and Automation, May, 1990.

[2] Cohen, A. D., "User Interface Requirements Document (DR SY45.1)," MDC H4261 (Revision A), McDonnell Douglas Space Systems Company-Space Station Division, March 1990.

[3] Dewberry, B. S., "The Environmental Control and Life Support System Advanced Automation Project - Phase I Application Analysis," Proceeding of the Space Operations Automation and Robotics Conference, June, 1989.

[4] Dewberry, B. S., "Automation of the Environmental Control and Life Support System," Proceedings of the NASA Space Station Evolution Symposium, February, 1990.

[5] Dewberry, B. S., "Space Station Freedom ECLSS - A Step Toward Autonomous Regenerative Life Support Systems," NASA CP-3073, Proceeding of the Fifth NASA Conference on AI for Space Applications, Huntsville, AL, May, 1988, pp. 193-201.

[6] Prince, N. R., et. al., "Challenges in the Development of the Orbiter Atmospheric Revitalization Subsystem," NASA No. 2342, Part 1, NASA Space Shuttle Technical Conference, June, 1983, pp. 465-479.

[7] Scarl, E. A., Jamieson, J. R., and New, E., "Deriving Fault Location and Control from a Functional Model," Proceedings of the 3rd IEEE Symposium on Intelligent Control, Arlington, VA, August, 1988.

[8] Scarl, E. A., Jamieson, J. R., and Delaune, C. I., "A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle." In Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, August, 1985, pp. 414-416.

[9] Scarl, E. A., Jamieson, J. R., and Delaune, C. I., "Monitoring and Fault Location at the Kennedy Space Center," in Newsletter of the ACM Special Interest Group on Artificial Intelligence (SIGART), No. 93, July, 1985, pp. 38-44.

[10] Scarl, E. A., Jamieson, J. R., and Delaune, C. I., "Sensor-Based Diagnosis using Knowledge of Structure and Function," IEEE Transactions of Systems, Man, and Cybernetics, SMC-17, No. 3, May/June, 1987, pp. 360-368.

[11] Shema, D. B. and Boose, J. H., "Refining problem-solving knowledge in repertory grids using a consultation mechanism," International Journal of Man-Machine Studies, Vol. 29, No. 1, July, 1988. pp. 447-460.

ADDRESSING INFORMATION OVERLOAD IN THE MONITORING OF COMPLEX PHYSICAL SYSTEMS

Richard Doyle
Jet Propulsion Laboratory

(Paper not provided at publication date)

THE CITS EXPERT PARAMETER SYSTEM (CEPS)

Arthur Nagel
Boeing

(Paper not provided at publication date)

EXPERT MISSILE MAINTENANCE AID (EMMA)

Capt. Barry Mullins
AFATL

(Paper not provided at publication date)

N91-20684

A Failure Management Prototype: DR/Rx

David G. Hammen
Carolyn G. Baker
Christine M. Kelly
Christopher A. Marsh

The MITRE Corporation
1120 NASA Road One
Houston, Texas 77058

Abstract

This failure management prototype performs failure diagnosis and recovery management of hierarchical, distributed systems. The prototype, which evolved from a series of previous prototypes following a spiral model for development, focuses on two functions: the Diagnostic Reasoner (DR) performs integrated failure diagnosis in distributed systems, and the Recovery Expert (Rx) develops plans to recover from the failure. This paper discusses issues related to expert system prototype design, discusses the previous history of this prototype, and describes the architecture of the current prototype in terms of the knowledge representation and functionality of its components.

Introduction

Space Station Freedom has been defined to have a hierarchical, distributed control architecture. The highest level in the architecture, Tier I, has knowledge of each of the systems in Freedom (for example, the Communications and Tracking System (C&TS) and the Thermal Control System (TCS)). The Operations Management System (OMS), composed of both automated functions and manual operations, represents Tier I in the command architecture. The second level, Tier II, represents a lower level in the command hierarchy, having a limited scope of knowledge (for example, the System Management function for the Electrical Power System, which has little or no knowledge of the other systems). Tier II managers' functions are further delegated to Tier III managers. The data become more abstract and qualitative as they advance upward through the control hierarchy.

This paper describes a prototype* that is being designed to perform failure management at the Tier I (OMS) level. Failure management includes diagnosing the failure, determining the corrective actions to take, and then taking the actions and tracking the progress of the recovery. The first phase of the project implements the first two of these three functions: the Diagnostic Reasoner (DR) performs diagnosis and the Recovery Expert (Rx) establishes a Course of Action to take to effect recovery.

This paper discusses issues related to expert system prototype design, discusses the previous history of the current prototype, and describes the architecture of this prototype in terms of the knowledge representation and functionality of its components.

Related Works

Our current effort expands on previous work done by others and by ourselves. Our current prototype expands on our previous efforts (Marsh, 1988; Marsh, 1989) by greatly increasing the use of behavior representation, by addressing the impacts that result from a failure, and by developing plans to recover from a failure. The Diagnostic Reasoner incorporates research from model-based reasoning, focusing on the works of Davis (Davis, 1985), de Kleer and Williams (de Kleer, 1987), Geffner and Pearl (Geffner, 1987), and Holtzblatt, Marcotte and Piazza (Holtzblatt, 1989). The Recovery Expert incorporates research from planning, focusing on the goal-directed planning developments by Wilkins (Wilkins, 1988) and the

* The research on and development of this prototype was jointly sponsored by the National Aeronautics and Space Administration, Johnson Space Center, under contract NAS9-18057, and by The MITRE Corporation under its MITRE Sponsored Research Program.

Procedural Reasoning System described by Georgeff and Ingrand (Georgeff, 1989).

Design Methodology

Design techniques used to build knowledge-based expert systems are quite different from those used to develop conventional software systems. Conventional software systems are developed using principles of modern software engineering, while expert systems development follows knowledge engineering disciplines.

Software systems developed using the waterfall model follow well-defined design methodologies and techniques and procedures that support them. This allows the project manager to control the software development process; the developer is provided a foundation for building high-quality software in a productive manner (Pressman 1987). NASA has baselined the use of the waterfall model for the development of software for the Space Station Freedom Program (NASA 1989).

A pitfall to avoid when following this method is its over emphasis on fully-elaborated documents in the early design phase at the expense of attention to functionality and meeting the user needs. The waterfall model is appropriate where budget and schedule are the primary concerns, but is ill-suited when good user interfaces and decision support aids functions are required (Boehm, 1988).

In developing a knowledge-based expert system, the phases of the development process are interleaved. The capabilities of the product evolve as a function of operating experience. This technique is well suited to knowledge-based applications where the concepts are not well known at the start of the project. This promises a rapid initial operating capability from which the product can evolve (Boehm, 1988). The key tasks for the developer are to gather domain knowledge from an expert, build a portion of the system, and then work with the expert to refine the product (Waterman, 1986).

A pitfall to avoid when using the iterative methodology is a tendency to incorporate additional capabilities that exceed the initial design assumptions and constraints; the resulting product is no longer an integrated piece of software but a large and unruly collection of routines and constructs. At this point, the design should be re-assessed and the system re-implemented to improve the conceptualization of the existing knowledge, if the system is to continue to grow in depth and breadth (Hayes-Roth, 1983).

The spiral model proposed by Boehm (Boehm, 1988) describes a development spiral in which concepts are discovered, implemented as prototypes and evaluated. The prototypes are discarded, but the valid concepts are retained and re-implemented in a more refined product. The spiral model provides for product life-cycle evolution and growth and focuses on identifying and resolving risk items.

History

The past history for the evolution of the OMS prototype has largely followed the spiral model, with ideas being re-implemented as operating concepts have matured. This paper describes the current phase in the prototype life cycle in which new ideas are being added, and some previous work is being re-implemented to reflect a closer-to-operations environment.

The first prototypes, implemented in a Lisp environment, demonstrated the use of inferencing in failure diagnosis and the use of automation in activity execution and monitoring. Eventually, the first prototypes were integrated on a test bed with simulation of Space Station Freedom systems (Marsh, 1988).

Once the test bed environment matured, it was necessary to refine on the capabilities of the first prototypes and re-implement them based on test bed operational constraints. A combination of C and Ada were used for this phase of implementation (Marsh, 1989; Kelly, 1989).

This paper describes the next step in the prototype evolution. An additional capability (Rx, to plan for recovery) is being added and integrated with DR, the failure diagnosis component. DR, a re-implementation of failure diagnosis, contains earlier diagnostic capabilities, but expands on the use of models, both to support diagnosis and to assist in the planning for recovery. Eventually, these two components will be integrated with the execution of the activities identified to effect recovery.

OMS Prototype Design

The OMS failure management prototype will be implemented in Ada (the language mandated for the Space Station Freedom Program) and ART/Ada (an expert system shell) on a VAX Station 3100 workstation under the VMS operating system.

We will continue using the spiral development methodology for this effort. Some iteration is required for the development of our knowledge and model bases as Freedom's design is subject to change. Iteration is also required for the development of our application software as the exact techniques to use or avoid are not yet well-known. The spiral methodology embodies this need for iteration.

The spiral method can support the complexity of the OMS design and provides the rigor necessitated by early definition of Ada specifications and interfaces and Ada's emphasis on strong typing. Compared to the very large projects developed using the waterfall model, the OMS prototype and development team are quite small; the extensive project management, configuration management, and documentation required by software engineering are not necessary for our small prototyping effort.

The design of two of the prototype's functions, the DR and Rx, has been recently completed. DR determines likely failure sources and their potential impacts and Rx develops plans to recover from these failures. An overview of the information flow through DR and Rx processes is presented in figure 1. This figure introduces a hypothetical scenario that relies in part on interactions between a C&TS frame multiplexer and a TCS cold plate; this scenario will be used to illustrate the design. Discussions of the design of these two functions follow.

DR Architecture

The DR is responsible for determining the likely source(s) of a failure and synthesizing dynamic summary failure reports from the Tier II systems. The DR's diagnosis could confirm or correct system-level diagnoses.

Updating the Component Model

The Tier II managers notify the DR of changes in the status of system components and changes in the relationships between those components through System Reports. Only significant qualitative changes

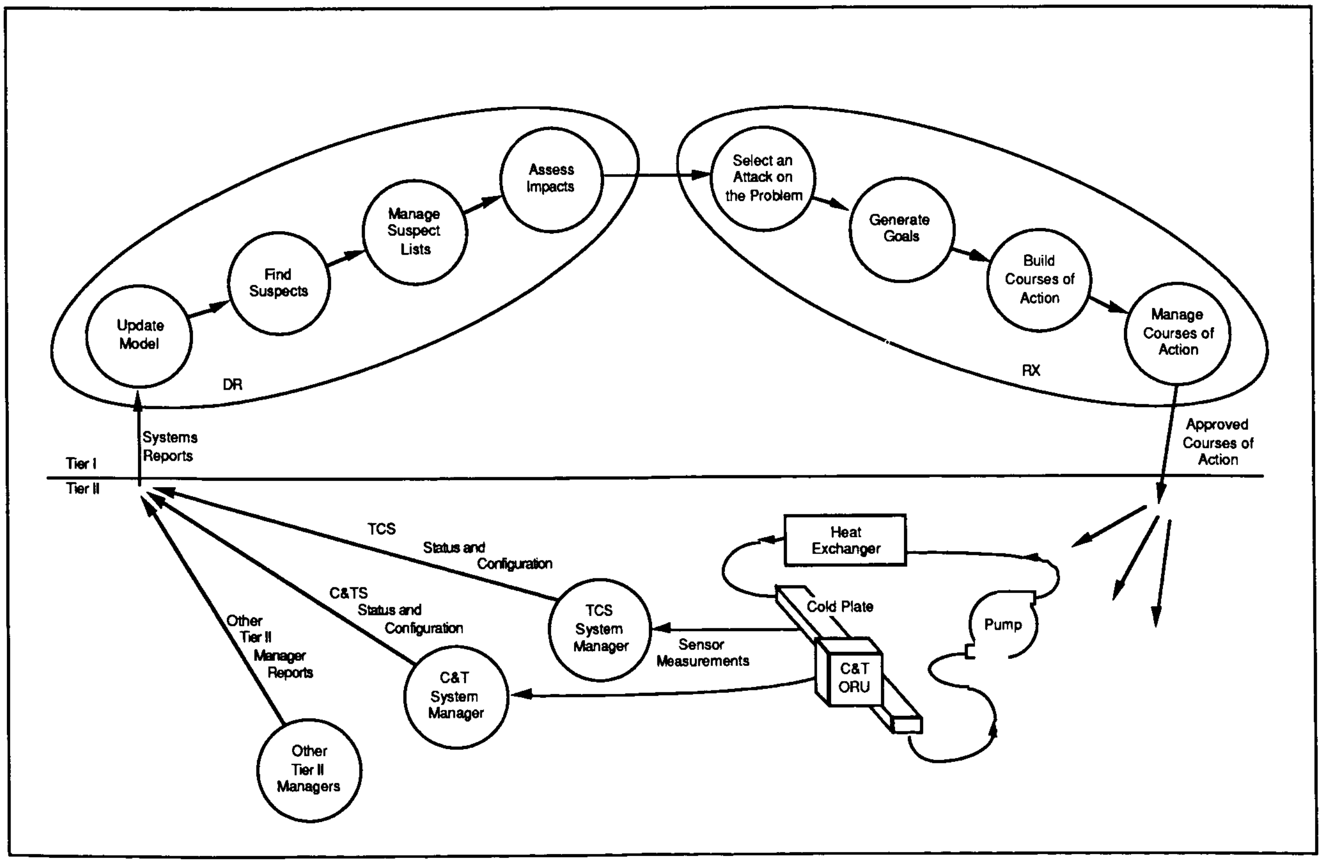


Figure 1 Overview of DR/Rx Process Flow

(for example, a cold plate's temperature changing from "nominal" to "hot") are reported by the systems to DR; minor quantitative deviations (for example, the cold plate's temperature changing from 70 to 72 degrees) are not. DR uses the information in the System Reports to update a schematic-like model of the Space Station Freedom's systems.

The component model incorporates configuration, status and behavior information. The configuration information identifies a component's relationships with other components, both physically and functionally. The status information identifies mode of operation, equipment health, and key operational measures that are related to behavior. The behavior information identifies the causes and effects of particular conditions with respect to a particular component's health and mode of operation. The behavior information describes both internal causal consequences and behaviors across configuration boundaries.

The description of a specific component is based on the generic description of a class of related components; the class descriptions in turn could be defined as a hierarchy of descriptions. The description of a class of components includes descriptions of behavior causes and effects and attribute definitions of behavior measure and configuration elements. The description of a specific component includes information about the component's behavior measures, operating conditions, and configuration. A portion of the Component Model is depicted in figure 2.

Generating a Suspect List

When the System Reports indicate a problem (as opposed to a nominal change in configuration), DR determines suspected causes based on reported behavior and modeled behavioral cause-and-effect. This information is collected into a Suspect List. DR verifies that the expected behavioral effects have occurred with respect to the possible suspects and their related components.

A Suspect List identifies those suspects that will result in a particular set of observed behaviors. More than one cause could result in same observable behaviors, in which case DR will identify each possible cause as a possible suspect in the same Suspect List. A set of observable behaviors could result from multiple component failures, in which case DR will identify the failure group as a possible cause. A Suspect List also could identify key unknown behavior measures: the assessment of some behavior measures will require special resources or will induce inter-system interactions. When DR encounters one of these unknown measures along one of its diagnostic causal pathways it will post the measure in the Suspect List as a key unknown behavior measure whose assessment should help refine the diagnosis.

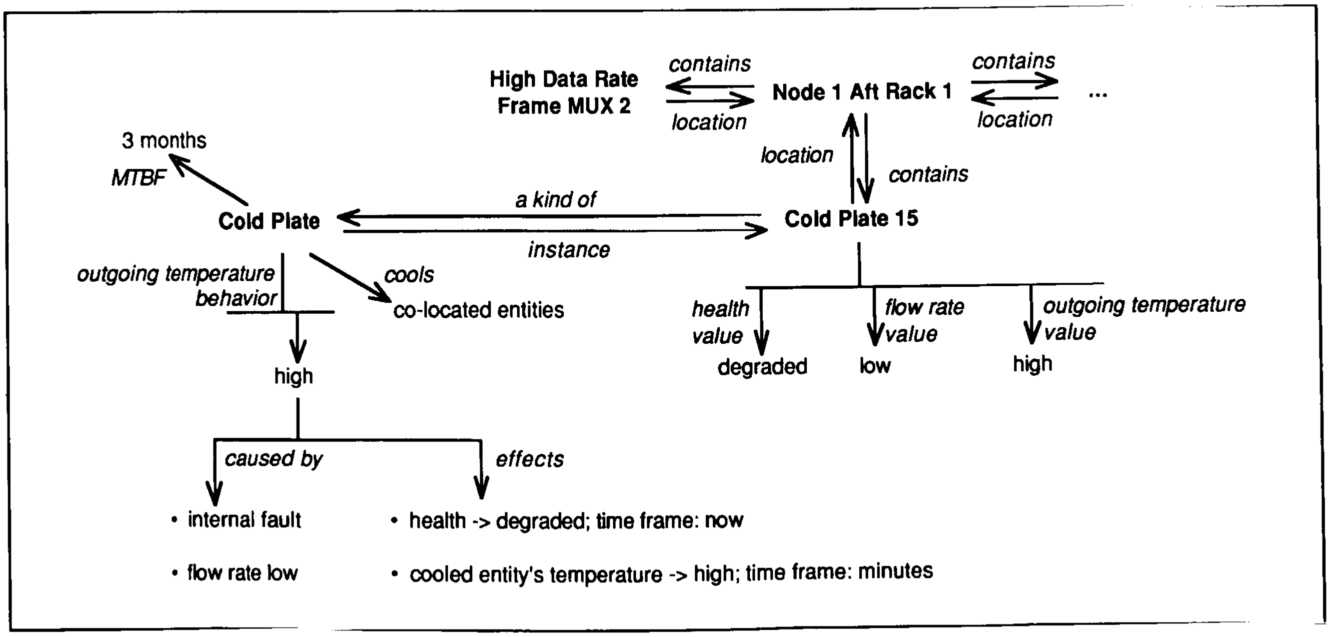


Figure 2 Component Model

Managing Suspect Lists

The Tier II managers do not observe all of the effects of a failure at one time. Consequently, DR will generate Suspect Lists without complete knowledge of the problem. When DR receives the first System Report, DR responds given the available information. As additional information becomes available, DR relies on the expected behavioral effects or explained behaviors of identified suspects that are described in the component model to merge Suspect Lists generated as the result of the same failure.

Assessing Impacts

The suspected components that produce the most immediate and most critical impacts should be considered before those suspects which have less severe consequences. To assess the impact of a particular suspect, DR uses an Impact Model that augments the Component Model. The Impact Model focuses on cascading operational causes and effects and looks further ahead in time than does the Component Model. To help assess the significance of an impact, the Impact Model heuristically assigns numerical severity values to an impact. A severity value and temporal factor are attached to each node in the derived, suspect-specific Impact Sequence. A sample impact sequence is presented in figure 3.

Rx Architecture

The Rx is responsible for determining and recommending a set of procedures, based on the available crew procedures, that will result in recovery from the problem. This set of procedures could include intermediate actions that mitigate the more acute consequences of the problem, providing adequate time to realize the recovery itself.

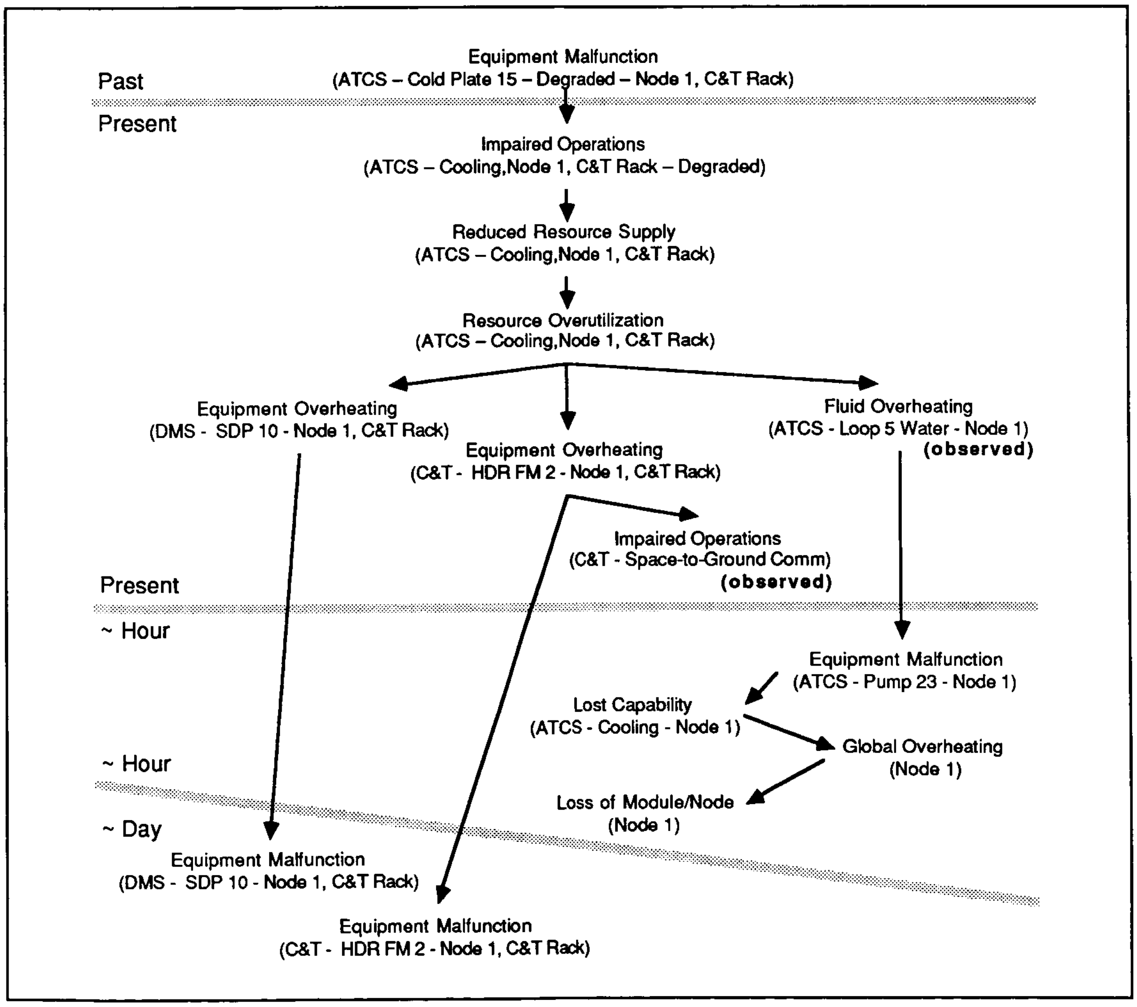

Past
Present
Present
~ Hour
~ Hour
~ Day
Equipment Malfunction (ATCS – Cold Plate 15 – Degraded – Node 1, C&T Rack)
Impaired Operations (ATCS – Cooling,Node 1, C&T Rack – Degraded)
Reduced Resource Supply (ATCS – Cooling,Node 1, C&T Rack)
Resource Overutilization (ATCS – Cooling,Node 1, C&T Rack)
Equipment Overheating (DMS - SDP 10 - Node 1, C&T Rack)
Equipment Overheating (C&T - HDR FM 2 - Node 1, C&T Rack)
Fluid Overheating (ATCS - Loop 5 Water - Node 1) (observed)
Impaired Operations (C&T - Space-to-Ground Comm) (observed)
Equipment Malfunction (ATCS - Pump 23 - Node 1)
Lost Capability (ATCS - Cooling - Node 1)
Global Overheating (Node 1)
Loss of Module/Node (Node 1)
Equipment Malfunction (DMS - SDP 10 - Node 1, C&T Rack)
Equipment Malfunction (C&T - HDR FM 2 - Node 1, C&T Rack)

Figure 3 Impact Sequence

Selecting an Attack

Several options are available for dispositioning the diagnosis reported by DR. Some failures are accurately identified by DR; in these cases, the problem can be addressed directly. Other failures are not easily identified; additional information is needed, such as information provided by an inter-system diagnostic test procedure or a behavior measure value that is unknown but whose assessment involves inter-system interactions (and therefore the value cannot be determined without Tier I approval). Rx is also concerned with assuring that the actions taken are sensible in light of the foreseeable impacts. When severe or acute downstream impacts could occur, Rx will develop plans that mitigate these downstream impacts so that the desired action can be sensibly performed. It might be imprudent for Rx to initiate any action when a preliminary Suspect List is reported by DR: Rx might do nothing until DR has observed some predicted near-term downstream impacts. Finally, Rx will request operator intervention if it cannot find an adequate response.

Generating Goals

Rx develops goals that address the failure in concert with the chosen attack. For example, repair goals directly address failures, impact mitigation goals address downstream impacts, data collection goals address unknown behavior measures, and diagnostic goals address unclear diagnoses. These goals will drive the generation of a plan to solve a specific part of the overall problem.

Rx applies generic goals that address the selected attack to the specific problem, forming a specific goal that addresses the specific problem and the selected attack. For example, Rx could address a reduction in cooling capacity as a special case of a resource supply reduction. The generic goal of reducing resource consumption addresses this generalized problem. Applying this generic goal is applied to the specific problem results in the specific goal of reducing the cooling load. Sample goal generation data for impact mitigation are presented in table 1.

Building Courses of Action

A Course of Action specifies a set of procedures that collectively achieve a specific goal. The procedures are selected from the pre-defined set of flight procedures. Procedure metadata describes reasons for, outcomes of and constraints against the use of procedures. Rx uses this procedure metadata to build a Course of Action that achieves the specific goals within the constraints imposed by the failure and its impacts. The Course of Action specifies the names of procedures and the temporal relationships between them: the procedures, when executed, should achieve the specific goal that the Course of Action addresses.

Managing Courses of Action

Rx can build multiple Courses of Action in response to a single problem. For example, the desired action should be achieved by a repair Course of Action, but several impact mitigation Courses of Action will be required for this desired action to have a successful outcome. These multiple Courses of Action must be merged and ordered to form a unified Course of Action that addresses the entire problem rather than a portion of the problem. The attempt is to build Courses of Action that solve the root problem within the constraints levied by the failure. Sample Courses of Action are presented in table 2.

Rx can also develop alternate means of addressing the problem. These alternate Courses of Action must be evaluated prior to execution. These final steps, as well as all other steps in the process, require crew interaction and approval.

Conclusions

The design of the DR and Rx has been recently completed. This design was achieved by using software engineering and knowledge engineering techniques. These techniques can be merged under the spiral model for a more integrated and long-lived system. This design incorporates some concepts from the predecessor prototyping activities but also introduces some new ideas.

Future Plans

A Procedures Interpreter was a previous product of the evolution of this prototype. This will be folded into our current work to execute the recommended Course of Action and to ensure that this Course of Action is achieving the desired goals. The current prototype is a stand-alone product and will be integrated into the test bed environment to demonstrate the effectiveness and an integrated failure management system.

Table 1
Goal Generation Information for Impact Mitigation

Impact	Workarounds	Goal Generation Information
Impaired Operations	Use Backup Capability	Backup Capability Mode On and Failed Equipment Mode Off
Reduced Resource Supply	Augment Resource Supply	Resource Level Nominal
Resource Overutilization	Augment Resource Supply or Decrease Resource Utilization	Resource Level Nominal Resource Consumption <= Resource Level

Table 2
Courses of Action

Goal	Course of Action	Comments
Repair Cold Plate	Repair Cold Plate 15	Severe impacts occur before completion
Reduce Cold Plate Load	Cross-Strap Frame Multiplexer 2	Does not attack root problem
Reduce Cold Plate Load and Repair Cold Plate	Cross-Strap Frame Multiplexer 2 and Repair Cold Plate 15	Timely and effective

References

Boehm, Barry W. (May 1988) A Spiral Model of Software Development and Enhancement. *IEEE Computer*, Volume 21, Number 5, pp 61-72.

Davis, Randall (1985), *Diagnostic Reasoning Based on Structure and Behavior*, Qualitative Reasoning About Physical Systems, Daniel G. Bobrow, ed, Cambridge, Massachusetts: The MIT Press, pp 347-410.

De Kleer, Johan and Brian C. Williams (1987), *Diagnosing Multiple Faults*, Artificial Intelligence, Volume 32, Number 1, pp 97-130.

Geffner, Hector and Judea Pearl (1987), An Improved Constraint-Propagation Algorithm for Diagnosis, Proceedings of the Tenth International Joint Conference on Artificial Intelligence, San Mateo, California: Morgan Kaufmann Publishers, Inc., pp 1105-1111.

Georgeff, Michael P. and François Felix Ingrand (1989), *Decision-Making in Embedded Reasoning Systems*, Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, San Mateo, California: Morgan Kaufmann Publishers, Inc., pp 972-978.

Holtzblatt, L. J., R. A. Marcotte and R.L. Piazza (October 1989), *Overcoming Limitations of Model-based Diagnostic Reasoning Systems*, presented at the AIAA Computers in Aerospace VII Conference.

Hayes-Roth, Frederick, Donald A. Waterman and Douglas B. Lenat, eds. (1983), *Building Expert Systems*, Reading, Massachusetts: Addison-Wesley Publishing Company Inc.

Kelly, Christine, Christopher Marsh, Alain Jouchoux and Fred Lacy (1989), *The Migration of an Expert System from Lisp to Ada*, Proceedings of AIDA-89: Fifth Annual Conference on Artificial Intelligence and Ada, Fairfax, Virginia: George Mason University, pp 108-119.

Marsh, Christopher (1988), *The ISA Expert System: A Prototype System for Failure Diagnosis on the Space Station*, Proceedings of the First International Conference on Industrial Engineering Applications of Artificial Intelligence and Expert Systems, New York, New York: ACM Press, pp 60-74.

Marsh, Christopher and Christine Kelly (1989), *Operations Management Application Prototype*, Fourth Artificial Intelligence and Simulation Workshop, Detroit, Michigan, pp 75-77.

Pressman, Roger S. (1987), Software Engineering: A Practitioner's Approach, New York, New York: McGraw-Hill.

NASA (February 1989), *Information System Life Cycle and Documentation Standards, Rel 4.3*, Washington D.C.: NASA Office of Safety, Reliability, Maintainability, and Quality Assurance–Software Management and Assurance Program.

Waterman, Donald A. (1986), *A Guide to Expert Systems*, Reading, Massachusetts: Addison-Wesley Publishing Company Inc.

Wilkins, David E. (1988), *Practical Planning: Extending the Classical AI Planning Paradigm*, San Mateo, California: Morgan Kaufmann Publishers, Inc.

N91-20685

AUGMENTATION OF THE SPACE STATION MODULE POWER MANAGEMENT AND DISTRIBUTION BREADBOARD

Bryan Walls, David K. Hall, and Louis F. Lollar
NASA, Marshall Space Flight Center
Bldg. 4487 EB12
Huntsville, AL 35812

ABSTRACT

The Space Station Module Power Management and Distribution (SSM/PMAD) Breadboard, located at NASA's Marshall Space Flight Center (MSFC) in Huntsville, Alabama, models power distribution and management, including scheduling, load prioritization, and FDIR, within a Space Station Freedom Habitation or Laboratory module. This 120 VDC system is capable of distributing up to 30 kW of power among more than 25 loads.

In addition to the power distribution hardware, the system includes computer control through a hierarchy of processes. The lowest level consists of fast, simple (from a computing standpoint) switchgear, capable of quickly safing the system. At the next level are local load center processors, called Lowest Level Processors (LLP's), which execute load scheduling, perform redundant switching, and shed loads which use more than scheduled power. Above the LLP's are three cooperating Artificial Intelligence (AI) Systems which manage load prioritizations, load scheduling, load shedding, and fault recovery and management. Recent upgrades to hardware and modifications to software at both the LLP (now based on 80386's) and AI System levels promise a drastic increase in speed, a significant increase in functionality and reliability, and potential for further examination of advanced automation techniques.

BACKGROUND

As the electrical power requirements for spacecraft have increased, the problems of managing these large power systems have also increased. America's first space station, Skylab, employed an eight kW power bus which required fifteen to twenty ground support personnel to monitor and control. Extensive crew involvement was also required at times to correct system faults. After the final crew left and Skylab was powered down, the EPS was evaluated . In the conclusion of this evaluation, ten recommendations for future spacecraft electrical power systems were presented. Seven of the ten recommendations can be implemented by the use of automation techniques. Based on these results and experience from other spacecraft EPSs, NASA/MSFC began to investigate automating a spacecraft EPS .

The first steps taken toward an automated EPS began in 1978 with the start of the The Autonomously Managed Power System (AMPS) program. The AMPS program was funded by NASA's Office of Aeronautics and Space Technology (OAST) and managed by MSFC through a contract with TRW. The AMPS program was a three phase program. The first phase identified a reference photovoltaic electrical power system for a 250 kW class low earth orbit (LEO) satellite. The second phase developed the autonomous power management approach for the reference EPS. The third phase developed a breadboard test facility to evaluate, characterize, and verify the concepts and hardware resulting from phases 1 and 2.

Based on the results of AMPS, a project to investigate automation techniques appropriate to a large PMAD system such as will exist on Space Station Freedom modules was begun in 1984 at MSFC. With the support of Martin Marietta Space Systems Group, the SSM/PMAD test bed was developed . Originally delivered as a 20kHz, 208V ring bus system, the SSM/PMAD power system has evolved with Space Station Freedom into its current 120VDC Star Topology. Development of the automation software

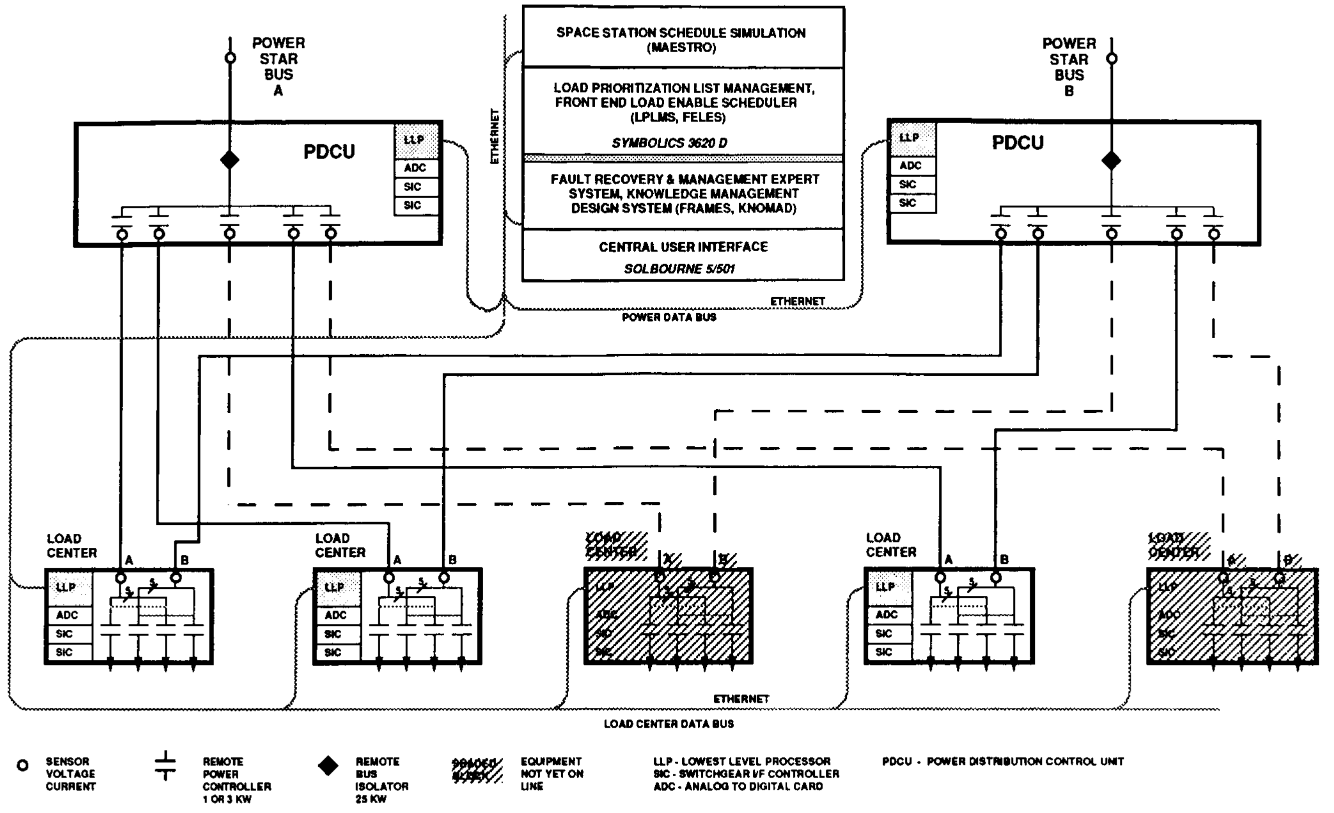

POWER STAR BUS A
POWER STAR BUS B
PDCU
LLP
ADC
SIC
SPACE STATION SCHEDULE SIMULATION (MAESTRO)
LOAD PRIORITIZATION LIST MANAGEMENT, FRONT END LOAD ENABLE SCHEDULER (LPLMS, FELES)
SYMBOLICS 3620 D
FAULT RECOVERY & MANAGEMENT EXPERT SYSTEM, KNOWLEDGE MANAGEMENT DESIGN SYSTEM (FRAMES, KNOMAD)
CENTRAL USER INTERFACE
SOLBOURNE 5/501
ETHERNET
POWER DATA BUS
LOAD CENTER
A
B
LOAD CENTER DATA BUS
SENSOR VOLTAGE CURRENT
REMOTE POWER CONTROLLER 1 OR 3 KW
REMOTE BUS ISOLATOR 25 KW
EQUIPMENT NOT YET ON LINE
LLP - LOWEST LEVEL PROCESSOR
SIC - SWITCHGEAR I/F CONTROLLER
ADC - ANALOG TO DIGITAL CARD
PDCU - POWER DISTRIBUTION CONTROL UNIT

Figure 1 -- SSM/PMAD Breadboard Topology

continues, but the system has already demonstrated fully autonomous operation, including scheduling, implementation of the schedule, and the ability to handle and diagnose several kinds of faults.

SSM/PMAD

The Space Station Module/Power Management and Distribution breadboard power system (Figure 1) is a two bus system, each consisting of a 120 Vdc power supply, one 15 kW Remote Bus Interrupter (RBI), five 3kW Remote Power Controllers (RPCs), and several 1 kW RPCs at the load center level. Each power bus is configured in a star bus arrangement with each RPC equipped with sensors that detect undervoltage, surge current, ground fault, high temperature, and I^2t conditions. If any of these conditions arise, the RPC will trip and then store the trip condition in its memory. Each RPC also provides switch status and data from a built-in current sensor. The system load banks are resistive and switchable in two 250 W increments, and one 500 W step to provide up to 1 kW of load to each RPC.

The system software is distributed through several types of processors. Processing at the level nearest the power hardware is performed by the Lowest Level Processors (LLPs). The LLPs are rack mounted 80386-based IBM/PC compatible computers with boards for Ethernet Local Area Network communications. Each LLP is responsible for controlling its associated switches and for monitoring all sensor readings and switch status in its center. The LLP also notifies the next higher machine, a Solbourne (Sun compatible) workstation, of any anomalies noted. Each LLP communicates down to one or two Switch Interface Cards (SICs), which communicate with the RPCs and the Analog to Digital Converter (A/D) Cards for sensor packets.

ORIGINAL PAGE IS OF POOR QUALITY

The Solbourne workstation contains the heart of the system software. The workstation houses the Fault Recovery and Management Expert System (FRAMES) and serves as the user interface for the breadboard in both manual and autonomous mode.

FRAMES monitors the system for anomalies. It receives the schedule from Maestro at the same time it is being sent down to the LLPs. Each LLP sends notification of any anomalies it sees, such as tripped switches or shed loads. Sensor reading messages are also sent to FRAMES. FRAMES uses the information which comes to it and attempts to find an explanation. If this explanation requires removing some pieces of equipment from service, FRAMES does so and notifies Maestro to adjust the schedule accordingly. FRAMES explains to the user the reasoning it followed, and shows on its system diagram screen the results of the fault or anomaly.

Maestro is a resource scheduler which can create a schedule based on multiple constraints. In the SSM/PMAD Breadboard the constraints currently used include number of crew members required, equipment resources, and power resources, with power being the resource of most concern in this breadboard. Power is allocated not just by the amount available to the system, but also by the ability of intervening components to supply the power. Maestro is housed on a Symbolics 3620D AI workstation.

The third system, the LPLMS, uses information from the event list and the active library, along with its own rules, to dynamically assign relative priority to each active load in the system. A new list is sent down to the LLP's at least every 15 minutes (less if a contingency occurs). The load priority list can be used to shed loads in case of a reduction in power.

INTERFACE TO LERC TEST BED

In an effort to share technology and demonstrate the effectiveness of advanced automation techniques, a joint project with Lewis Research Center (LeRC) has been initiated. An interactive link will be forged between the LeRC power system automation test bed and the SSM/PMAD system. A simple scenario to demonstrate basic functionality between the two is scheduled for the end of July 1990, with a demonstration due by the end of November. A more robust demonstration will be planned using the knowledge gained from these initial tests for the July 1991 time frame.

As currently envisioned, this project will consist of a link, via Ethernet, between the two breadboards. The link will operated at two levels: the power system level and the control level. At the power system level, the SSM/PMAD system will appear as a load to the LeRC test bed, and the LeRC system as a source to SSM/PMAD. This will be accomplished by letting the LeRC control one (and, ultimately, both) of the power supplies of the SSM/PMAD system, while the SSM/PMAD system will control the characteristics of a programmable load attached to the LeRC system. At the control level, information will go back and forth between the two systems negotiating power requirements versus available resources for the scheduling systems, and notification of any faults or contingencies for the FDIR functionality.

The initial demonstration will show only the control link, using simple negotiation algorithms.
Functionality will be added as the interconnection becomes more robust.

LASEPS

The AMPS high power DC breadboard was mentioned in the introduction as a precursor to SSM/PMAD. AMPS still exists, and shares the lab with SSM/PMAD. The Large Autonomous Spacecraft Electrical Power System (LASEPS) is an in-house effort to combine the strengths of both systems, resulting in a single source to load power breadboard. AMPS' contribution will be to replace the existing 120 Vdc power supplies with its two power channels, thus giving the system a solar array/battery network power source more representative of an actual flight power system. Each power channel will have its own Solar Array Simulator and 108 cell, 189 Ampere hour Ni-Cd battery supplying a 145 +\- 15 Vdc bus.

A Programmable Power Processor (P^3) will be used on each bus as an interface between AMPS and SSM/PMAD. These P^3's, developed in the late 1970's for a 25 kW power module, are microprocessor controlled voltage regulators with an input voltage range up to 400 Vdc, and an output range from 24 to 180 Vdc at a maximum of 100 A. These specifications qualify the P^3's as a convenient and efficient interface between the two breadboards.

This complete, complex, high-power breadboard will enable the MSFC team to further investigate techniques and criteria required for any large space power system such as will be required for the Lunar/Mars initiative, large platforms envisioned for Mission to Planet Earth, or other new agency goals, all without compromising the function of the SSM/PMAD system which is a part of it.

FUTURE PLANS

As SSM/PMAD matures, more and more effort will go into making sure the technology being developed is used. This includes identifying the necessary impacts to the design, development, and operation of Space Station Freedom for implementation of the technology at both Permanent Manned Compatible (PMC) and Assembly Complete (AC) stages. Work continues with Boeing to support WP01 SSF tasks, encouraging use of the breadboard where possible. Demonstrations of the breadboard's capabilities will be as broad-based as possible, including personnel from the other NASA centers, their prime contractors, and Headquarters.

Efforts will also continue to make the technology more amenable to actual flight use. Code will be converted to Ada, starting at the lowest levels and moving up. Functionality will continue to be moved to as low a level as possible to take advantage of the power of distributed computing. A study into the possibility of using microcontrollers to perform some or all of the LLP and lower functionality is now underway. The demonstrations with LeRC will help to point out weaknesses of both systems as part of a larger system. LASEPS will be of similar value. Development of the user interface will help in showing the true power of the system, and will actually increase that power by making it more accessible.

The system is already capable of autonomous operation. An important basic addition to the breadboard will be intermediate modes of autonomy, enabling a user to take "semi-manual" control -- that is, being able to modify the system without having to take full manual control. This will result in a much safer, more robust system than is possible with only fully autonomous or fully manual operation.

REFERENCES

This paper includes, with some modification, large sections of the first two references below. The other references are included for further reading about the system.

David K. Hall and Louis F. Lollar, "Development of an Automated Electrical Power Subsystem Testbed for Large Spacecraft." Proceedings of the 25th IECEC, pub. pending, August 1990.

Louis F. Lollar and Bryan Walls, "Automating Large Electrical Power Systems for Spacecraft"

Barry R. Ashworth. "An Architecture for Automated Fault Diagnosis." Proceedings of the 24th IECEC. Volume 1, pp.195-200, August 1989.

Barry R. Ashworth. "Managing Autonomy Levels in the SSM/PMAD Testbed." Proceedings of the 25th IECEC. pub. pending, August 1989.

Barry R. Ashworth. "Reactive Autonomous Planning in Spacecraft.." Proceedings of the Aerospace Applications of Artificial Intelligence Conference, 1989.

Daniel L. Britt, John R. Gohring, and Amy L. Geoffroy, "The Impact of the Utility Power System Concept on Spacecraft Activity Scheduling." Proceedings of the 23rd IECEC, Vol. 3, pp. 621-626, August 1988.

Rajiv Doreswamy, "Implementation of a Virtual Link Between Power System Testbeds at Marshall Spaceflight Center and Lewis Research Center." Proceedings of the 25th IECEC, pub. pending, August 1990.

Kenneth A. Freeman, Rick Walsh, and David J. Weeks, "Concurrent Development of Fault Management Hardware and Software in the SSM/PMAD." Proceedings of the 23rd IECEC, Vol. 3, pp. 307-312, August 1988.

Roy Lanier, Jr., Robert Kapustka, and John R. Bush, Jr.: "A Programmable Power Processor For a 25-kW Power Module." NASA Technical Memorandum 78215, January 1979.

S.C. Lee and Louis F. Lollar, .: "Development of a Component Centered Fault Monitoring and Diagnosis Knowledge Based System for Space Power Systems." Proceedings of the 23rd IECEC, Volume 3, August 1988.

L.F. Lollar and D.J. Weeks, "The Autonomously Managed Power Systems Laboratory." Proceedings of the 23rd IECEC. Volume 3, pp.415-419, August 1988.

J.W. McKee, Norma D. Whitehead, and Louis F. Lollar, .: "Considerations in the Design of a Communication Network for an Autonomously Managed Power System." Proceedings of the 24th IECEC, Volume 1, August 1989.

William D. Miller, and Ellen F. Jones, "Automated Power Management Within a Space Station Module." Proceedings of the 23rd IECEC, Vol. 3, pp. 395-399, August 1988.

William D. Miller, et al. "Space Station Automation of Common Module Power Management and Distribution – Interim Final Report." NASA Contractor Report 4260, November, 1989.

Regina Palmer, "A Paradigm for Testing Embedded Expert Systems", Eighth Annual Pacific Northwest Software Quality Conference, pub. pend, October 1990.

Joel D. Riedesel. "Diagnosing Multiple Faults in SSM/PMAD." Proceedings of the 25th IECEC, pub. pending, August 1990.

Joel D. Riedesel. "An Efficient, Distributed Architecture for Cooperating Expert Systems." Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90), pub. pending, August 1990.

Joel D. Riedesel. "Knowledge Management: An Abstraction of Knowledge Base and Database Management Systems." NASA Contractor Report 4273, January 1990.

Joel D. Riedesel. "Knowledge Management: An Abstraction of Knowledge Base and Database Management Systems." Proceedings of the Fifth Annual AI Systems in Government Conference, May 1990.

Joel D. Riedesel. "A Survey of Fault Diagnosis Technology." Proceedings of the 24th IECEC. Volume 1, pp.183-188, August 1989.

Joel D. Riedesel, Chris J. Myers, and Barry R. Ashworth. "Intelligent Space Power Automation." Proceedings of the Fourth IEEE International Symposium on Intelligent Control, 1989.

Joel D. Riedesel and Bryan Walls, "A Knowledge-Base Architecture for Distributed Knowledge Agents." Proceedings of the Fifth Conference on Artificial Intelligence for Space Applications, pp.241-256, May 1990.

Bryan Walls, "Exercise of the SSM/PMAD Breadboard." Proceedings of the 24th IECEC. Volume 1, pp.189-194, August 1989.

Bryan Walls and Louis F. Lollar, "Automation in the Space Station Module Power Management and Distribution Breadboard." Proceedings of the Third Annual Workshop on Space Operations Automation and Robotics (SOAR '89), pp.107-112, July 1989.

David J. Weeks and Bryan Walls, "Space Station Freedom Advanced Development Project Plan for SSM/PMAD Automation." Internal NASA memo for Contract NAS8-36433.January 1990.

REAL-TIME SYSTEMS CONTROL

PRECEDING PAGE BLANK NOT FILMED

N91-20686

The Real-Time Control of Planetary Rovers Through Behavior Modification

David P. Miller
Jet Propulsion Laboratory/California Institute of Technology
M.S. 301-440, 4800 Oak Grove Drive
Pasadena, CA 91109
(818) 354-9390
dmiller@ai.nasa.jpl.nasa.gov

Abstract

It is not yet clear of what type, and how much, "intelligence" is needed for a planetary rover to function semi-autonomously on a planetary surface. Current designs assume an advanced AI system that maintains a detailed map of its journeys and the surroundings, and that carefully calculates and tests every move in advance. To achieve these abilities, and because of the limitations of space-qualified electronics, the supporting rover is quite sizable, massing a large fraction of a ton, and requiring technology advances in everything from power to ground operations.

An alternative approach is to use a behavior driven control scheme. Recent research has shown that many complex tasks may be achieved by programming a robot with a set of behaviors and activating or deactivating a subset of those behaviors as required by the specific situation in which the robot finds itself. Behavior control requires much less computation than is required by traditional AI planning techniques. The reduced computation requirements allows the entire rover to be scaled down as appropriate (only down-link communications and payload do not scale under these circumstances). This paper discusses the missions that can be handled by the real-time control and operation of a set of small, semi-autonomous, interacting, behavior-controlled planetary rovers.

1. Introduction:

There are many possible uses for unmanned planetary rovers. Rovers with a high degree of autonomy can carry out missions that serve science, operations, and space exploitation goals. For example, rovers can be used on the Moon to perform site certification for possible manned outposts and science instrument sites. On Mars, science instruments need to be placed and soil and rock samples need to be gathered from a wide variety of terrains. To reduce light-time delays and the need for communications (and its inherent infrastructure of relay satellites etc), rovers with at least semi-autonomous capabilities, are highly desired.

1.1 Plan Control for Rovers

The autonomous system control that has been proposed for a rover, to accomplish the tasks mentioned above, is shown in Figure 1. The rover senses its environment, combines that with previous knowledge (from earlier and orbital views) and then builds a map of its surroundings. A path planner finds a trajectory through the map. The trajectory is simulated, producing run-time expectations, and these expectations are monitored during the actual rover traverse. If an expectation is violated, the rover performs a reflex stop, and starts the cycle over again. Under normal circumstances the cycle is repeated every five to fifteen meters. Such a system has been successfully implemented, and tested under realistic conditions [Miller89, Gat90].

The implemented system required just under one billion machine instructions per meter of travel. While the code used in this experimental system was by no means optimal, by the time all the functionality and reliability improvements are made, it is believed that the real number will be within a factor of three. This means that a rover that needed to travel at a speed of one kilometer per hour would need a (space qualified) computational capability of between 80 and 250 MIPS. Some of this could be offloaded onto special purpose computation systems, but none the less, computation becomes a major driver for a planetary rover, both in power and mass.

The system studies that have been undertaken [Pivirotto90] have confirmed this, indicating that a rover with some onboard autonomy would need to mass between 600 and 842kg (the test vehicle we used massed over 1100kg). In large part, this mass is due to the system control algorithms.

1.2 Low-Mass Rovers are Needed

For almost all imagined uses of rovers, the job is best done if it can be done several times, in many places, under differing conditions. Traditional rovers weigh several hundred kilograms, and under the best circumstances will probably be able to move a few kilometers a day. It will not be economically feasible to place more than a few such rovers on a planetary surface. Nor will it be logistically possible to have those few rovers visit all the desired sites. Additionally, the risk of losing one of these scarce and expensive resources will make it difficult to put a rover in the places where it could do the most good: in the previously unmapped or geologically unknown areas of a planetarv surface.

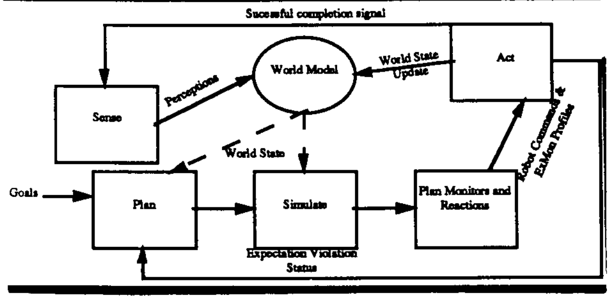

Sucessful completion signal
World Model
World State Update
Act
Perceptions
Sense
World State
Robot Commands & ExMon Profiles
Goals
Plan
Simulate
Plan Monitors and Reactions
Expectation Violation Status

Figure 1. Planning Cycle

All of the problems mentioned above go away if a low-computation method for autonomously controlling rovers could be developed. Low computation greatly lowers the power needs of the rover, which can greatly reduce its mass. Once the power mass cycle is broken, it is possible to scale the entire rover. The strength to weight ratio of mechanical systems improves as a system is scaled smaller. This in turn makes the system stronger and stiffer, which will reduce the accuracy of the control needs, further reducing the power and payload needs of the rover, allowing it to be scaled even smaller. The rover cannot be made arbitrarily small, because it is necessary for it to carry some sort of payload, but improvements in instrument miniaturization [Waltman89] indicate that that may not be much of a limitation. A rover must also travel through its environment, but remember that an ant can travel anywhere a person can, and many more places in addition.

To keep this all within todays technology, if one could build mini-rovers that massed a few kilograms, and could operate autonomously, then many useful things could be done with them. Mini-rovers could be sent in far greater numbers, and to potentially far riskier sites. It now appears feasible to design a small rover that can accomplish most of the science and operations objectives that previously would have been handled by a large rover. Key in building a mini-rover is behavior programming.

2. Behavior Control for Rovers

Behavior programming is a methodology that allows a set of interacting reactions to be programmed into a robot so that they work together. For example, if a robot has a behavior that causes the robot to turn so as to maximize the value being returned by its right side range finder, and has a behavior to cause the robot to turn away when its left side proximity sensor is activated, then that robot will exhibit the behavior of traveling around the perimeter of a room in a clockwise direction.

Some behavior languages have been created [Brooks86, Gat90], and several robots have been programmed with these languages to perform some interesting tasks [Brooks89, Gat90]. Central to having a robot perform something of interest is the ability to have the robot's behavior modified by cues in the environment, or by the robot's own actions.

Figure 2 shows the sensors and actuators of a small robot named "Tooth". Tooth is a robot massing just under two kilograms, and containing two eight-bit micro-processors and four kilobytes of memory. Its memory is filled with the behavior program shown in Figure 3. Tooth's mission is to go around a room, picking up "toys" that have been left near the walls of the room, and to bring the toys to a beacon located somewhere near the center of the room. The robot carries out this task using nine interacting behaviors. The look for toy behavior just has the vehicle move along in a straight line. The follow light behavior uses the photocells to decide on the direction of the beacon. If the beacon can be seen, this behavior sends a steering command away from the beacon. This forces the robot towards the walls. The dead-end, obstacle avoidance, unthrash, and stall behaviors all keep the robot from crashing into anything, or getting stuck. When the object beam sensor detects something, then the robot uses the pickup toy behavior to grasp the object. If it is successful, then that behavior causes the steering signal from the follow-light behavior to be inverted. This causes the robot to head towards the beacon. When the photocells report a light above a certain threshold brightness, then the drop toy behavior turns on. This overrides the look for toys behavior, causes the robot to stop, drop the toy and back away a few centimeters. Since the robot is no longer holding the toy, the follow-light signal is no longer inverted, and the robot turns away from the light and goes in search of more toys. [Gat90] gives more details on these experiments.

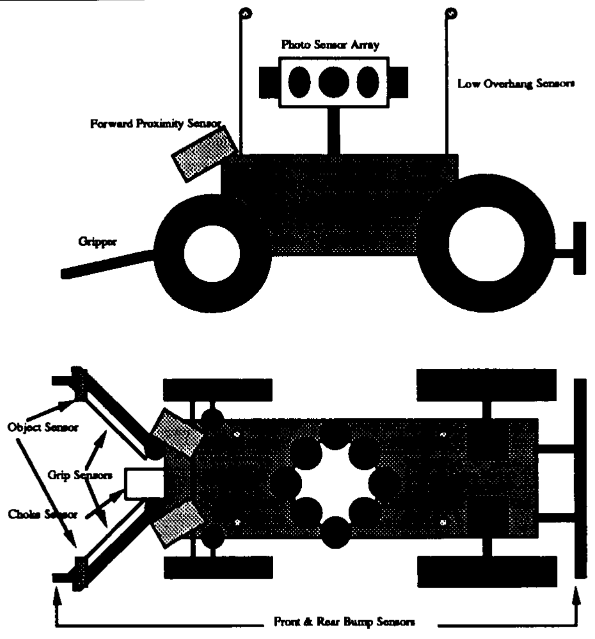
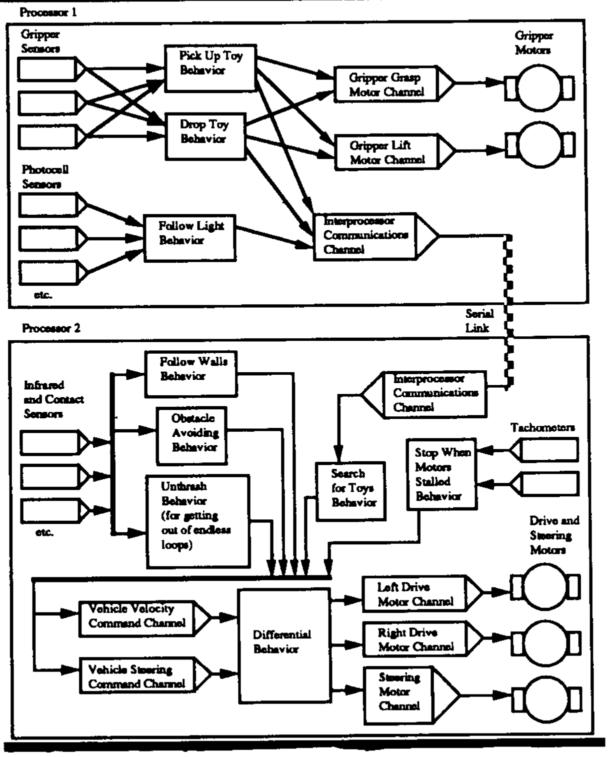


Figure 2: The Tooth Robot

The experiment outlined above shows that simple behaviors can be linked together to exhibit complicated actions. The experiment above had the robot perform all the major parts of a sample return mission. The robot moved following certain parameters. When it came across something that matched its sample criteria, it acquired it. It then brought the sample back to a marked beacon (simulating a return vehicle dock). All the while, the robot avoided obstacles in its path. This technology is sufficient for a variety of missions.

3. Target missions:

There are many missions that can be performed by behavior-controlled robots. The advantages of these small, autonomous rovers are many. Because of their small size, low mass, and high strength, these rovers could be placed on a planetary surface without the expense and mass of a traditional soft lander. [Miller90] outlines how many mini-rovers could be landed on Mars using parachutes and areoshells, and how communications can be maintained through ground relays. On the Moon, mini-rovers could be landed using an updated version of the Ranger seismometer capsule [Ranger63]. Navigation can be handled by referencing the robot to a coded radio signal, such as that used in VOR aviation radios.



Processor 1
Gripper Sensors
Pick Up Toy Behavior
Drop Toy Behavior
Gripper Grasp Motor Channel
Gripper Lift Motor Channel
Gripper Motors
Photocell Sensors
Follow Light Behavior
Interprocessor Communications Channel
etc.
Serial Link
Processor 2
Infrared and Contact Sensors
Follow Walls Behavior
Obstacle Avoiding Behavior
Unthrash Behavior (for getting out of endless loops)
etc.
Interprocessor Communications Channel
Search for Toys Behavior
Tachometers
Stop When Motors Stalled Behavior
Drive and Steering Motors
Vehicle Velocity Command Channel
Vehicle Steering Command Channel
Differential Behavior
Left Drive Motor Channel
Right Drive Motor Channel
Steering Motor Channel

Figure 3: The Behavior Program for Tooth

3.1 A Lunar Mission

On the Moon, one of the major activities of rovers would be deploying science instruments (e.g., VLFA). The VLFA is large antenna consisting of several hundred elements laid out over a sixty kilometer long spiral arc. The exact placement is not crucial, but the elements should be distributed evenly along the arc. The elements themselves are self contained and mass approximately one kilogram. A previous study undertaken by the Battelle Corporation [Easter88] chose to use 1060kg rover that could carry the approximately 300kg of antenna elements. The rover contained large robotic arms for implanting the antenna elements. Since the VLFA is to be placed on the Lunar Farside, a series of relay

ORIGINAL PAGE IS
OF POOR QUALITY

satellites would be placed in lunar orbit to allow near continuous communications with the rover.

There are several ways the VLFA could be set up using behavior controlled mini-rovers. The simplest method would be to build one antenna element into each mini-rover, and land approximately 300 on the lunar farside. It would be easy to have the rovers distribute themselves relative to one another, in the desired pattern. This, most inefficient use of mini-rovers, would still result in a considerable mass savings over the Battelle proposal.

One could also send up, say fifty mini-rovers, that could each retrieve antenna elements from a central cache. It would then be trivial to have the rovers place the antenna elements in the proper arrangement about a VOR transmitter located at the cache. Each rover would have prestored the coordinates (radial and range) of the six antenna elements it was to place. Simple behaviors would cause the rovers to home in on the transmitter till they could spot an antenna element. Once they picked that up, they would circle the transmitter till they located the proper radial. They would then head outward along the radial the proper distance and deposit the antenna element. No longer holding anything, they would head back towards the cache and repeat the process with the next set of position coordinates. This scheme would require only about 500kg landed on the Moon, much lower lander masses, and only the relay satellites required by the VLFA.

Any activity that needs to take place in a pattern can easily be done with behavior-controlled mini-rovers. A simple non-directional radio-beacon along with a range encoder can be used to have a rover go in circles, spiral in or out, or rendezvous at the beacon. By adding the radial information that comes from a VOR, a mini-rover could have its behaviors direct it to a specific point relative to the beacon. All the while, other behaviors can be used to keep the rover from hanging up on obstacles, or getting stuck in dead-ends.

3.2 A Mars Mission

On Mars, there are several unanswered questions which require rovers to work at many diverse locations. In particular, the search for carbonates, and the emplacement of science stations (seismic and meteorological) are very suitable for behavior-controlled mini-rovers. These tasks require rovers in many different terrains in areas that are many thousands of kilometers apart, and are most useful in areas where the rover may never be able to get out (eg., inside extinct volcano craters). Here, the basic philosophy should be put down a lot of rovers, all over the place, and have them report back when they find something interesting.

More narrow scope missions on Mars are also applicable. A question, which could be key for the design of a sample return mission, is the thickness of the weathering rind on Mars rocks. Scientific opinion ranges from a millimeter to several centimeters. The extent of the rind will determine the type of coring that will be necessary for a sample return mission. A small rover, armed with a small circular saw, would be able to answer this question (or at least determine if the rind is more than a centimeter). Such a rover would have behaviors to get it close up to rocks of various sizes. It would have a list of criteria, as soon as it found a rock matching some of those, it would come up to the rock, and using the control behaviors, cut off a slice. It would then take several images and relay them to Earth, then go off to find its next sample. With the proper cameras, such a rover could gather information about the weathering rind, and about the makeup of many Mars' rocks.

The weathering rind mission should be done very soon. Such a mission could be done very economically. Two or three mini-rovers would be dropped to the planet's surface via parachutes. A small relay orbiter would be left in orbit. Each rover would find an appropriate sample, take the images, and send a signal to the orbiter. When the orbiter was in range, it would broadcast a ready to receive signal, and the rover could dump the images directly from the camera CCDs to the orbiter. This way, the rovers need no mass storage, and can get by with very simple electronics.

The information from these images is crucial in designing a proper rover for a detailed science sample return mission. If the weathering rind is several centimeters then a rover large enough to do rock coring is necessary for many experiments. If the rind is on the order of a millimeter, then a sample return mission using behavior-controlled mini-rovers (see [Miller90]) would be more than adequate, and much more cost-effective.

4. Conclusions

Controlling a rover through behavior-control can allow such a rover to be greatly reduced in size and complexity with little or no loss in capability. Planetary rover missions, in most cases, are rover missions because the mission objectives require the sensors to be in many different locations. All planetary missions are mass constrained. Behavior controlled mini-rovers are a way of getting beyond the mass constraints and increasing the effective mobility of the system. Mobility is increased by making the rovers more autonomous (detailed direction from Earth does not fit well with the behavior-control paradigm) and by being able to provide more rovers, dropped at more locations, for a given mass allotment, then is possible using traditional control or planning techniques. Behavior-control, when combined with nano-technolgy, also holds the promise of being able to undertake wholly new types of missions [Brooks89].

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.

References

[Brooks86] Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal on Robotics and Automation, vol RA-2, no. 1, March 1986.

[Brooks89] Brooks, R.A., Flynn, A.M., Fast, Cheap, and out of Control: A Robot Invasion of the Solar System, *Journal of the British Interplanetary Society*, vol 42, #10, pp478-485, October 1989.

[Easter88] Easter, D.A., Buoni, C.M., McCauley, L.A., Mobility study for a Lunar rover, in the *Proceedings of the Mobile Robots III Conference*, SPIE Cambridge Symposiums, SPIE vol 1007, pp128-135, November 1988.

[Gat90] Gat, E., Slack, M.G., Miller, D.P., Firby, R.J., Path Planning and Execution Monitoring for a Planetary Rover, in *Proceeding of the IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May 1990.

[Gat90b] Gat, E., Miller, D.P., *BDL: A Language for Programming Reactive Robotic Control Systems*, JPL Working paper, 1990.

[Miller89] Miller, D.P., Atkinson,D.J., Wilcox,B., Mishkin,A.H., Autonomous Navigation and Control of a Mars Rover, *Proceedings of the 11th IFAC Symposium on Automatic Control in Aerospace*, pp127-130, Tsukuba, Japan, July 1989.

[Miller90] Miller, D.P., Mini-rovers for Mars Exploration, in the *Proceedings of the Vision 21 Workshop*, NASA-Lewis Research Center, Cleveland, OH, April 1990.

[Pivirotto90] Pivirotto, D.S., Dias, W.C., *United States Planetary Rover Status - 1989*, NASA Jet Propulsion Laboratory Publication JPL 90-6, 1990.

[Ranger63] "Final Technical Report, Lunar Rough Landing Capsule Development Program," Aeronutronic Division Publication Number U-2007, February 1963.

[Waltman89] Waltman, S.B., Kaiser, W.J., Electron Tunnel Sensor Technology, *Journal of the British Interplanetary Society*, vol 42, #10, pp474-477, October 1989.

QUALITATIVE CHARACTERIZATION FOR REAL-TIME CONTROL AND MONITORING

Steven Margolis
Aerospace Corporation

(Paper not provided at publication date)

N91-20687

Reasoning about Real-Time Systems with Temporal Interval Logic Constraints on Multi-State Automata*

Armen Gabrielian

Thomson-CSF, Inc.
630 Hansen Way, Suite 250
Palo Alto, CA 94304

Abstract

Models of real-time systems using a single paradigm often turn out to be inadequate, whether the paradigm is based on states, rules, event sequences or logic. In this paper, a model-based approach to reasoning about real-time systems is presented in which a temporal interval logic called TIL is employed to define constraints on a new type of high-level automata. The combination, called "Hierarchical Multi-State (HMS) machines," can be used to model formally a real-time system, a dynamic set of requirements, the environment, heuristic knowledge about planning-related problem solving, and the computational states of the reasoning mechanisms. In this framework, mathematical techniques have been developed for (1) proving the *correctness* of a representation, (2) *planning* of concurrent tasks to achieve goals, and (3) *scheduling* of plans to satisfy complex temporal constraints. HMS machines allow reasoning about a real-time system from a model of *how* truth arises instead of merely depending on *what* is true in a system.

1. Introduction

Real-time systems are characterized by unpredictability of inputs and "hard deadline" requirements. In addition, since many real-time systems are utilized in life-critical situations, strict "safety properties" are usually defined for them. A safety property is a state of affairs that must always remain true in a system. Instead of the usual discussion of "liveness properties," it is useful to define other requirements of a real-time system in terms of a set of "conditional goals" defined in terms of (condition, goal) pairs. A condition defines the state of affairs under which the associated goal must be pursued. We assume that deadlines may be associated with goals and that requirements are dynamic so that the pursuit of an active goal may have to be abandoned if certain other conditions become true. Thus, at the specification stage, the main forms of reasoning about a real-time system consists of the *verification* that (1) safety properties are not violated and (2) conditional goals are achievable. For traditional systems which operate deterministically or stochastically, this is essentially sufficient even though it can be a very complicated process. At the operational stage, two other forms of reasoning arise for "intelligent systems" which are not defined deterministically and require a search or other forms of analysis to instantiate a specific set of responses in a particular situation. First, off-line reasoning can be performed to determine in advance a set of allowable actions to achieve goals. Secondly, on-line reasoning can be employed, where deadlines on the reasoning process itself may have been defined. A key problem in the specification and operation of complex real-time systems is the choice of a representational framework that can provide manageable approaches to specification, verification, and instantiation of behavior.

While numerous formal representational schemes have been proposed for systems in general and real-time systems in particular, most of these are based on one of the following paradigms: state-based

* The work reported in this paper was supported in part by the Office of Naval Research under Contract N00014-89-C-0022.

models, rules, event sequences or logic. Two major examples of state-based models are automata and Petri nets. For real-time systems, traditional automata are inadequate for at least two important reasons: (1) explosion of the state space for non-trivial systems, and (2) absence of a natural mechanism for representing temporal constraints. Petri nets reduce the state space and can represent concurrent activities adequately. However, in Petri nets numerous dummy states are usually necessary to maintain logical consistency and no clear separation is made between *precedence* and *causality* [7]. In addition, even timed Petri nets have a limited language for representing temporal relationships among states and events [6]. Specification and verification of complex real-time requirements are also difficult for rule-based systems and event sequences. In particular, it is generally accepted that while rules are appropriate for defining prototypical behavior, they are inadequate for reasoning about novel situations. As far as pure logical formalisms are concerned, temporal logic provides a promising approach, except for two shortcomings. First, certain simple regular properties cannot be expressed in temporal logic [10]. Secondly, in a pure logic-based language a system is represented merely in terms of *what* is true. This gives a limited understanding of system behavior, since knowledge of *how* truth arises which is common to state models is not readily available.

The purpose of this paper is to present a brief overview of a comprehensive framework for specifying real-time systems and reasoning about them, called "Hierarchical Multi-State (HMS) machines," that integrates high-level "multi-state" automata and fragments of a temporal interval logic called TIL ([7], [4], [3], [5], [6]). As noted in Figure 1, an HMS machine can be used to define formally the dynamic behavior of a system, its requirements, a model of the environment, heuristic knowledge about planning-related problem solving, and the state of the computational resources used in reasoning. Given such a specification, the system can be simulated, its correctness can be verified formally, and it can be used for both off-line and on-line reasoning to derive operational plans and schedules to respond to the dynamics of a real-time situation.

Section 2 presents an outline of a simple form of HMS machines, with a brief discussion of the method for representing requirements in terms of "policy HMS machines." Section 3 presents an overview of the planning process, plan representation languages and a scheduling algorithm for plans. Section 4 presents a brief set of conclusions and directions for future work.

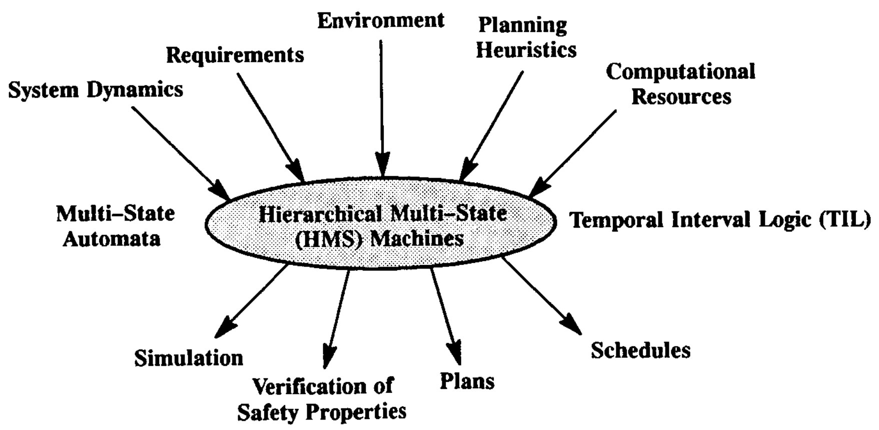


Figure 1. Specification, Verification and Reasoning Framework for Real-Time Systems

2. Automata, Temporal Logic, Machines and Real-Time Systems

An automaton consists of a set of "states" and a set of "transitions" that cause changes in states due to the occurrence of certain events such as arrival of inputs. This provides a very general architecture for defining the dynamics of a system, except that, as indicated in the Introduction, it is inadequate for specifying complex real-time systems. Hierarchical Multi-State (HMS) machines [4] are high-level "multi-state automata," in which (1) multiple (hierarchical) states can be true at one moment, (2) multiple transitions can fire simultaneously, and (3) a temporal interval logic, called TIL, is used to define constraints on transitions. This architecture allows the compact definition of the dynamics of complex real-time systems, in which interactions among states and hard deadlines can be defined formally. In addition, a "multi-level" combination of HMS machines [5] provides the capability for formally defining dynamic *requirements*, giving rise to a model-based reasoning framework for real-time systems. Because of limitations of space, only the non-hierarchical version of HMS machines will be considered here. A formalization of hierarchies can be found in [6].

An HMS machine is a triple $H = (S, \Gamma_D, \Gamma_N)$, where S is a set of "states," Γ_D is a set of "deterministic" transitions, and Γ_N is a set of "nondeterministic" transitions. Boolean states represent properties that may be true or false about a system. Non-boolean states can represent both properties of multiple entities in a system and properties of data objects. Deterministic transitions denote *fixed* causal interactions among states, while nondeterministic transitions represent *possible* or *permissible* interactions. Nondeterminism, in fact, is the key to the specification of *choice* in model-based reasoning in the HMS framework.

The constraints or "controls" on transitions in an HMS machine are defined in terms of the temporal interval logic TIL which is obtained by adding the following three operators to propositional logic:

$O(t)$:	*At* relative time t
$[t_1, t_2]$:	*Always* between times t_1 and t_2
$<t_1, t_2>$:	*Sometime* between times t_1 and t_2

The operators $[t_1, t_2]$ and $<t_1, t_2>$, which allow hard real-time constraints to be defined for HMS machines, are generalizations of the standard temporal logic operators $\Box$ and $\Diamond$, respectively. All times are relative, with the current moment denoted by 0. Figure 2 depicts a simple 2-level example of an HMS machine specification that defines both a nondeterministic "basic machine" H_1 and a specification of requirements in terms of the "policy HMS

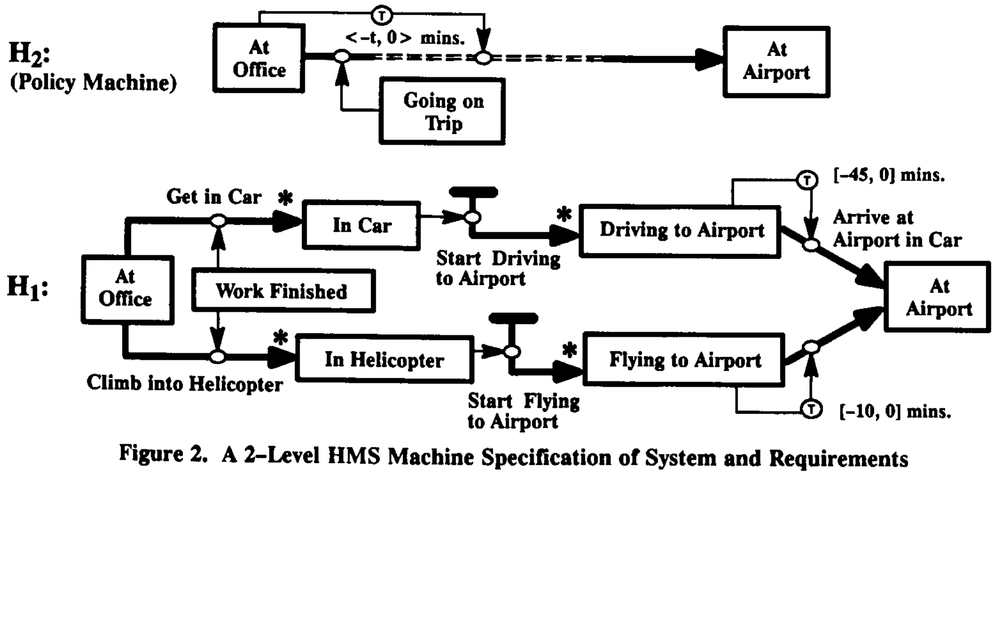


Figure 2. A 2-Level HMS Machine Specification of System and Requirements

machine" H_2. In this figure, rectangular boxes represent states, dark arrows are transitions, thin arrows denote TIL controls on transitions with the symbol $\textcircled{T}$ next to each temporal operator, and the partially double-dashed arrow in H_2 is a "policy transition" that defines intentionality. Asterisks denote nondeterministic transitions so that in H_1 the choice of all actions is not completely determined. We say that a transition is "enabled" if (1) its "primary" states from which the transition emerges are true, and (2) its controls are true. Thus, starting at the left side in the machine H_1, from the state "At Office" one can go into state "In Car" or into state "In Helicopter" as long as the control state "Work Finished" is true. If the state "In Car" ("In Helicopter") is true, then nondeterministically the transition "Start Driving to Airport" ("Start Flying to Airport") is fired. Nondeterminism is useful since this machine may be part of the specification of a much larger set of behaviors that could include going to many other destinations. The horizontal bar from which the transition "Start Driving to Airport" arises is an infinite resource which is always true. Thus, if this transition fires, both the states "In Car" and "Driving to Airport" would be true simultaneously. We note that at the end of this path, if the state "Driving to Airport" has been true continuously from 45 minutes earlier to the current moment, then a deterministic transition will take one to the state "At Airport."

The policy transition of machine H_2 in Figure 2 defines the goal of reaching the state "At Airport" when executing H_1, with the requirement that the state "Going on Trip" must be true in the beginning and the trip should not take more than t minutes. Thus, depending on the value of t, different "plans" for H_1 can be derived to reach the goal state. If the execution of the plan takes more than t minutes, then the plan can essentially be abandoned. Additional types of controls on policy transitions that are not shown in the figure can be used to define complex interactions of states and goals, including the capability of making a goal dependent on the planning process itself. Thus, for example, an alternate goal can be specified if the plan generation process takes longer than a specified length of time. Heuristic knowledge about plans can be captured by intermediate policy machines that define midpoint states that must be achieved during the execution of a plan. More details about policy machines can be found in [5].

An important benefit of the formal specification of a real-time system is that it provides a framework for verification of correctness and consistency before implementation. Following the procedure in 3, given an HMS machine and any safety property defined on its states, one can create a new "extended" state that will be true if and only if the safety property is violated. By a result of [8], such a state need only depend on the past history of the states of the machine, even though safety properties are usually defined in terms of future events. Two specific verification methods can then be used to verify that the extended state corresponding to the safety property is not reachable. In the first method, correctness-preserving transformations [3] are applied to modify an HMS machine incrementally, without affecting its behavior, until the safety state is isolated. In the second method, a "model-checking" approach [6] is used to demonstrate in finite time the correctness of infinite behavior. As in [2], this involves a branching simulation process that terminates paths when cycles are detected. A major advantage of using HMS machines is that orders of magnitude reduction in the number of states can be obtained in many applications compared to traditional automata models.

3. Planning, Plan Formalisms and Scheduling of Plans for HMS Machines

A "plan" in the HMS framework consists of a sequence of sets of transitions to be executed in a nondeterministic machine [5]. Conditional goals are specified for an HMS machine in terms of policy transitions of a policy HMS machine such as H_2 in Figure 2. The "planning" process then consists of searching the space of eligible nondeterministic transitions in a basic machine such as H_1 to derive a plan that causes the goal states of a policy transition to be reached. The important points to note in this framework are that (1) goals can be defined formally in terms of histories of states that are being modified dynamically, (2) circumstances such as inability to meet a deadline may cause a goal to be dropped from consideration, (3) the states of the computational resources in which planning is being performed may be used as controls on the policy transitions that define goals, and (4) heuristic guidelines for deriving plans can be specified in terms of intermediate policy machines.

Compilation of plans in advance to meet goals with hard deadlines has been proposed by number of authors (see, e.g., [9]). Various representation schemes for plans have also been proposed. For example, in [1] a Petri net model is used to define conditional actions that depend on facts that are true about the environment. The HMS machine framework offers a powerful capability to define complex concurrent plans that depend not only on the current states of the world but also on temporal histories of states. For this purpose, we say that a machine P is a "plan HMS machine" for a nondeterministic machine H, if some of the states of P correspond to the nondeterministic transitions of H and some other states are "dependent" states of the states of H. A dependent state is defined as a state for which (1) truth only depends on a logical combination of the truth or falsehood of other states, and (2) there are certain restrictions on transitions emerging from it and entering it. At each moment of time, the "execution" of P on H then is obtained by (1) firing the transitions of P as in a standard HMS machine, (2) firing the deterministic transitions of H, and (3) firing those nondeterministic transitions of H that are enabled in H and for which a corresponding state in P is true. Thus, for example, the plan machine in Figure 3 describes how the nondeterministic transitions in the machine H_1 should be executed. The states containing asterisks are dependent states which, in this case, are simply duplicates of corresponding states in H_1, assuming that the state "In a Hurry" is added to H_1. The states denoted by dashed rectangles represent *transitions* in H_1. Thus, this machine indicates that in case the state "In a Hurry" is true, one should execute the transition "Climb into Helicopter" from the state "At Office" in H_1. On the other hand, if the state "In a Hurry" is false, the transition "Get in Car" should be executed. Also, when the state "In Car" becomes true in H_1, the transition "Start Driving to Airport" will be fired if its corresponding state in Figure 3 is true. The latter situation will be true if the state "Going on Trip" has been true sometime earlier.

Two simpler formalisms for defining HMS machine plans can be defined in terms of the plan languages PL_0 and PL_1, which can also be considered as languages for describing concurrent event sequences. Words in the language PL_0 simply consist of sequences of (1) symbols from the set of transitions of the HMS machine, (2) lists of symbols, (3) words with integer exponents. An individual symbol α denotes the firing of the corresponding transition in the machine. A list of the form $(\alpha, \beta, \ldots, \delta)$ denotes the simultaneous firings of the transitions $\alpha, \beta, \ldots, \delta$. A word of the form w^n represents the n-fold repetition of firing of the transitions in w. Thus, the plan $\alpha\ (\beta, \gamma)\ (\delta\eta)^n$ denotes the execution of the fol-

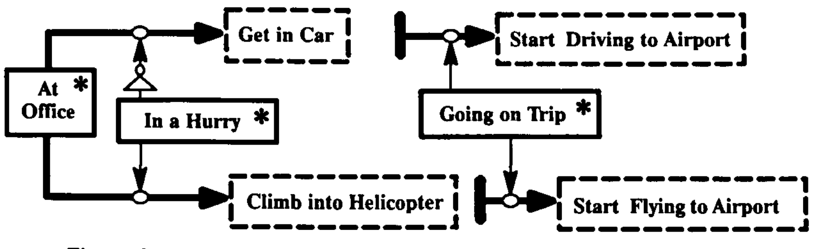


Figure 3. A Plan Machine for the HMS Machine H_1 in Figure 2.

lowing transitions in a machine: first fire α, then fire β and γ simultaneously, then fire δ followed by η n times. The language *PL_1* extends *PL_0* by the introduction of conditional operators and the means for defining alternative choices of actions. Plans in such languages, combined with an underlying HMS machine, provide the capability for both model-based reasoning from basic principles and the ability to respond rapidly to dynamic requirements without the need for searching.

Plan languages also offer the possibility of studying the *scheduling* of plans as distinct from the planning or plan generation process itself. For example, consider the plan

"Get in Car" "Start Driving to Airport" "Arrive at Airport in Car"

in the plan language *PL_0* for the machine H_1 of Figure 2. This plan simply lists the sequences of actions that must be performed, in which there is a key missing element: *when* should the actions be performed. Here, the only missing part is a delay of 45 minutes that must occur between the transitions "Start Driving to Airport" and "Arrive at Airport in Car." If such required delays are incorporated into a plan and it is verified for correctness, then the underlying machine can essentially be ignored during the execution. The important correctness criteria for plans are: (1) no transition is attempted that is not enabled, and (2) the plan will transform the machine from a given initial set of states to the desired final set of goal states.

In [5] a general approach for deriving schedules for plans was introduced that also provides a limited method of verifying the correctness of plans. In this scheme, given a potential plan p', a "variable delay plan" p is generated in which between each pair of terms in p' a parametric delay ϕ^i is introduced, where ϕ denotes a wait or "no action." Using symbolic execution techniques, then a solution for the exponents of the ϕ's can often be found that guarantees the correctness of the plan. In addition, in many cases, misordered plans can be corrected in the process of finding the delays.

4. Conclusions and Future Work

Hierarchical Multi-State (HMS) machines provide a framework for specification, verification and control of complex real-time systems by integrating multi-state automata and temporal interval logic. The major benefits are: (1) significant reduction in state space, (2) convenient mechanisms for specifying both safety properties and conditional goals, including hard deadlines, (3) methods of verifying correctness of specifications, and (4) model-based reasoning approaches for planning and scheduling in dynamic environments.

Three directions for future work have been defined: theory, applications and tools. Theoretical research goals include (1) the extension and formalization of the specification language, (2) investigation of more powerful methods for capturing requirements, (3) verification methods, (4) representation of uncertainty relating to both incomplete knowledge about the world and probabilistic outcome of events, (5) introduction of learning, and (6) efficient planning and scheduling algorithms. Various potential application areas for HMS machine have also been identified. Currently, HMS machines are being applied to the specification of a fragment of a future European command and control system. As far as tool development plans are concerned, work is continuing on the development of a prototype environment for specification of HMS machines, along with the capabilities for interactive simulation, limited forms of animation, and verification.

Acknowledgments

The author would like to acknowledge the contributions of Matthew K. Franklin in the development of the concepts discussed in this paper. Also, Ramachandran Iyer participated in the formalization of the temporal logic concepts discussed here and made helpful comments on an earlier draft.

References

[1] Drummond, M., "A representation of action and belief for automatic planning systems," *Proc. 1986 Workshop on Reasoning About Actions and Plans*, Morgan Kaufmann, 1987, pp. 189–212.

[2] Clarke, E.M., E.A. Emerson, A.P. Sistla, "Automatic verification of finite-state concurrent sytems using temporal logic," *ACM Transactions on Programming Languages and Systems*, Vol. 8, No. 2, 1986, pp. 244–263.

[3] Franklin, M.K., and A. Gabrielian, "A transformational method for verifying safety properties in real-time systems," *Proc. 10th Real-Time Systems Symposium*, Santa Monica, CA, Dec. 7–9, 1989, pp. 112–123.

[4] Gabrielian, A., and M.K. Franklin, "State-based specification of complex real-time systems," *Proc. 9th Real-Time Systems Symposium*, Huntsville, Dec. 10–12, 1988, pp. 2–11.

[5] Gabrielian, A., and M.K. Franklin, "Multi-level specification and verification of real-time software," *Proc. 12th International Conf. on Software Engineering*, Nice, France, March 26–30, 1990. To be reprinted in *Communications of the ACM.*

[6] Gabrielian, A. and R. Iyer, "Integrating automata and temporal logic: a framework for specification of real-time systems and software," *Proc. Unified Computation Laboratory*, Institute of Mathematics and its Applications, Stirling, Scotland, July 3–6, 1990, to appear.

[7] Gabrielian, A., and M.E. Stickney, "Hierarchical representation of causal knowledge," *Proceedings of WESTEX–87 IEEE Expert Systems Conference*, 1987, pp. 82–89.

[8] Manna, Z., and A. Pneuli, "The anchored version of the temporal framework," in *Proc. Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science 354, Springer-Verlag, 1989, pp. 201–284.

[9] Schoppers, M.J., "Representation and automatic synthesis of reaction plans," *Ph.D. Thesis*, Computer Science Dept., Univ. of Illinois, Urbana–Champaign, 1989.

[10] Wolper, P., "Temporal logic can be more expressive," *Information and Control*, Vol. 1, No. 1–2, 1983, pp. 72–99.

N91-20688

A Framework for Building Real-Time Expert Systems

S. Daniel Lee

Inference Corporation
550 N. Continental Blvd.
El Segundo, CA 90245

Abstract

NASA's Space Station Freedom is an example of complex systems that require both traditional and AI real-time methodologies. It has been mandated that Ada should be used for all new software development projects. The Station also requires distributed processing. Catastrophic failures on the Station can cause the transmission system to malfunction for a long period of time, during which ground-based expert systems cannot provide any assistance to the crisis situation on the Station. This is even more critical for other NASA projects that would have longer transmission delays (e.g. the Lunar base, Mars missions, etc.) To address these issues, we propose a distributed agent architecture (DAA) that can support a variety of paradigms based on both traditional real-time computing and artificial intelligence. The proposed testbed for DAA is APEX (Autonomous Power EXpert), which is a real-time monitoring and diagnosis expert system for the electrical power distribution system of NASA's Space Station Freedom.

1. Introduction

The current, ongoing work of Inference, the "Real-Time Expert Systems" project for NASA Johnson Space Center, under a subcontract to the University of Houston - Clear Lake, has provided valuable insights into requirements for real-time knowledge-based systems being developed for NASA's Space Station Freedom. NASA's Space Station Freedom is an example of complex systems that require both traditional and AI real-time methodologies. The standard on-board processor on the Station is an 80836-based workstation with limited memory. In the ground-based control center, on the other hand, conventional engineering workstations can be used for AI applications. It has also been mandated that Ada should be used for all new software development projects.

The Station also requires distributed processing. For example, if expert systems for fault detection isolation and recovery (FDIR) for the Station were fielded only in the ground-based control center, communication delays could cause serious problems. Catastrophic failures on the Station can cause the transmission system to malfunction for a long period of time, during which ground-based expert systems cannot provide any assistance to the crisis situation on the Station. This is even more critical for other NASA projects that would have longer transmission delays (e.g. the Lunar base, Mars missions, etc.)

However, current real-time knowledge-based system architectures suffer from a variety of shortcomings:

- A heavy dependence on inefficient implementation platforms, usually Common Lisp, which makes it difficult if not impossible to be deployed in real-time embedded systems.
- A weak integration with traditional real-time computing methodologies.
- An inability for the architectures to be distributed among multiple heterogeneous platforms that communicate asynchronously.

We have, previously, implemented an Ada-based expert system tool, ART-Ada, to facilitate the deployment of expert systems in Ada, which addresses the first point above [13], [14], [11], [15].

We propose a distributed agent architecture (DAA) that can support a variety of paradigms based on both traditional real-time computing and artificial intelligence.

2. Distributed Agent Architecture

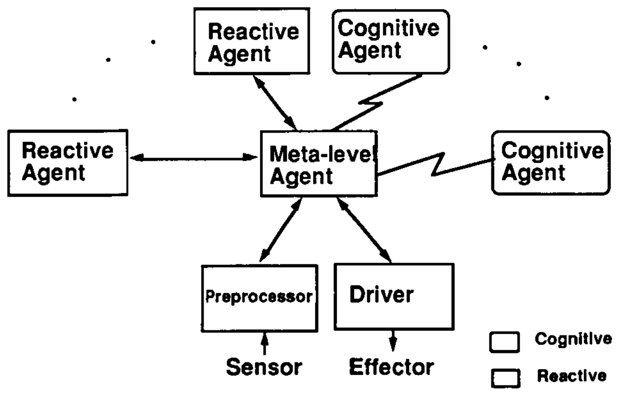


Figure 2-1: Distributed Agent Architecture

The distributed agent architecture (DAA) for real-time knowledge-based systems is depicted in figure 2-1. DAA has the following technical objectives:

- The overall system performance should satisfy real-time requirements. Onboard systems should prevent catastrophic failures during the absence of assistance from ground-based systems due to the malfunction of communication systems.

- Onboard systems should adapt gracefully to dynamic environments by trading quality for speed of response.

- The architecture should be based on distributed and cooperative processing, which will enable migration of knowledge-based system modules from ground-based systems to onboard systems.

- Its baseline implementation language should be Ada. Ada will make it possible to employ traditional real-time computing methodologies and to deploy knowledge-based systems in embedded systems. If both ground systems and onboard systems are implemented in Ada, it would be easier to migrate modules from ground to the Station.

DAA consists of distributed agents that are classified into two categories: reactive and cognitive. Reactive agents can be implemented directly in Ada to meet hard real-time requirements and to be deployed on on-board embedded processors. A traditional real-time computing methodology under consideration is the rate monotonic theory that can guarantee schedulability based on analytical methods [20], [21]. AI techniques under consideration for reactive agents are approximate or "anytime" reasoning that can be implemented using Bayesian belief networks as in Guardian [8], [7]. Fuzzy logic [16], [26], [22] and reactive planning [1], [5], [10], [17], [18] are also being considered for reactive agents.

Cognitive agents are traditional expert systems that can be implemented in ART-Ada to meet soft real-time requirements. During the initial design of cognitive agents, it is critical to consider the migration path that would allow initial deployment on ground-based workstations with eventual deployment on on-board processors. ART-Ada technology enables this migration while Lisp-based technologies make it difficult if not impossible.

In addition to reactive and cognitive agents, a meta-level agent would be needed to coordinate multiple agents and to provide meta-level control. An important area of coordination is timeline management. Following [2], we intend to implement three timelines --- occurred, expected, and intended --- where each timeline records one type of information. Any agents can process or post *events* in any timelines through the meta-level agent.

3. Reactive Agents

Reactive agents are designed to meet hard real-time requirements. Hard real-time requirements are different from soft real-time in that if hard deadlines are not met, catastrophic failures are likely to occur. Catastrophic failures include the loss of human lives, the loss of major hardware components, etc. On the other hand, even if soft deadlines are violated, no major catastrophic failures are likely to occur.

It is also critical that reactive agents fit into embedded processors of the Space Station Freedom. Some AI tasks can be directly implemented in a procedural language such as Ada. The use of Ada will enable us to take advantage of recent progress that has been made in the area of real-time computing in Ada. A noteworthy example is the rate monotonic theory that can guarantee schedulability based on analytical methods [20], [21].

The rate monotonic theory guarantees schedulability of multiple tasks if certain conditions are satisfied. There are some restrictions, however:

- The execution time of a task must be known because it is a parameter in conditions that must be satisfied.

- It assigns the highest priority to a periodic task with the shortest period. Therefore, it prevents tasks from having priorities based on other criteria.

- The theory applies only to multiple tasks --- periodic and aperiodic --- that reside on a single processor.

It is not clear whether the theory can be used for dynamic scheduling. It is usually used before the program execution to determine whether deadlines could be met. If deadlines are not met, periods of periodic tasks must be adjusted properly. We believe that the theory can be used to adjust periods dynamically if they are allowed to change dynamically. The theory does not prescribe how to find periods that would meet the deadlines, however.

With the right Ada runtime executive that supports rate monotonic scheduling, the schedulability can be guaranteed in advance by applying the theory analytically. It is expected that the Ada 9X Project will incorporate the rate monotonic algorithm in the next revision of the Ada language, which is due for release in 1993.

An AI technique that is useful for reactive agents is approximate or "anytime" reasoning. For example, Guardian uses a Bayesian belief network to provide reactive diagnosis. Each node of a Bayesian belief network is associated with an action. When a deadline is reached, Guardian simply recommends the action associated with the current node. If more time is given, it will continue to refine its belief and may recommend a conflicting action later on. We plan to implement an approximate reasoning module based on Bayesian belief networks in Ada.

Fuzzy logic-based systems [16], [26], [22] can also be used as reactive agents, using either modeling software or fuzzy hardware. In fact, fuzzy logic may subsume probabilistic reasoning using Bayesian belief networks. Fuzzy systems are becoming popular in Japan [19]. Togai InfraLogic, Inc in Irvine, California manufactures fuzzy-system chips and modeling software written in C. Fuzzy systems are suitable for reactive agents because:

- Real-time response can be achieved by implementing the logic on a chip.

- Fuzzy logic allows approximate reasoning.

Various reactive planning methods have been proposed [1], [5], [10], [17], [18]. These planning methods (a.k.a. universal planning) have been sharply criticized mainly for the exponential growth of their size with the complexity of the domain [6]. We plan to study both sides of arguments and investigate the possibilities of implementing reactive planning agents using some of these methods in DAA.

4. Cognitive Agents

Cognitive agents are traditional knowledge-based systems that are designed to meet soft real-time requirements. AI problems such as diagnosis demand accuracy of solution within a soft deadline rather than sacrifice of solution quality to meet a hard deadline. While reactive agents address the latter through approximate reasoning, cognitive agents should be based on AI techniques that facilitate deeper reasoning. For example, in Guardian, model-based reasoning is used for cognitive diagnosis while a Bayesian belief network is used for reactive diagnosis.

Although AI systems usually run on a ground-based engineering workstation today, it is becoming increasingly important that these systems are readily available in real-time embedded environments.

Inference has already developed ART-Ada, an Ada-based expert system tool, for this specific purpose. ART-Ada supports rule-based reasoning as well as frame-based reasoning that can be used to implement model-based reasoning. When the current version of ART-Ada is used, the total memory requirement for an ART-Ada application with hundreds of rules is 2-3 megabyte. It may be reasonable for embedded systems based on newer processors such as the Intel 80386 and 80960, the Motorola 68000 and 88000, and the MIPS RISC chip. It is important, however, to note that the current version of ART-Ada is not optimized. The primary focus of the current release was to provide functionality. Inference plans to release an optimized version of ART-Ada in the near future.

Because of numerous bugs found in the Ada compilers used for this project, we could not make some of the obvious performance optimizations that could have made ART-Ada faster and smaller [11]. In addition to compiler problems, we also discovered some fundamental issues with the Ada language itself that also affected the performance of ART-Ada [11]. In particular, the problem with dynamic memory management has the most significant impact on the execution size and performance of ART-Ada.

Our current research effort is focused on implementing ART-Ada's own memory manager using an existing technology. If it is not possible to implement it in Ada, we will implement it in an assembly language. Another area of research is to improve real-time support in ART-Ada. Several extensions to ART-Ada are proposed to address real-time issues and included in Appendix I.

5. A Meta-Level Agent

In a distributed architecture like DAA, the problem is how to provide meta-level control and coordination between distributed agents. A meta-level agent is a common blackboard for meta-level control and coordination. Some examples of meta-level control are:

- to control the data input rate of the preprocessor --- when a serious problem arises, the input data rate can be reduced so that agents spend more resources in dealing with the current situation;

- to assign tasks to agents --- crisis situations may have to be handled by reactive agents to provide quick fixes while cognitive agents may follow up on it later;

- to reconcile conflicting recommendations --- when reactive agents and cognitive agents make conflicting recommendations, it is necessary to reconcile the differences; and

- to schedule operations for effectors --- when multiple agents try to control effectors, it is necessary to schedule effector assignments.

Another important area of coordination is timeline management. Following [2], we intend to implement three timelines where each timeline records one type of information. The *occurred* timeline is used for representing facts acquired from monitoring sensors. The *expected* timeline represent what we expect in the future. The *intended* timeline represents *goals*. The intended timeline is different from the expected timeline in that actions can be taken to ensure that goals are met, whereas no actions need to be taken to produce expected results. Any agents can process or post *events* in any timelines through the meta-level agent. We intend to use ART-Ada to implement the meta-level agent.

6. Interagent Communication

There are several possible layers in the interagent communication protocol:

- protocol for interprocess communication,

- protocol for telemetry,

- protocol for distributed objects,

- protocol for distributed knowledge bases, and

- protocol for distributed autonomous agents.

Unix interprocess communication protocol (e.g. sockets and TCP/IP) would be a reasonable low-level protocol for prototypes. We intend to develop a protocol for distributed objects because we believe that it is an optimal layer for interagent communication. Other higher-level protocols are interesting research topics, but they may not be as practical as the distributed object protocol. Eventually, protocols used in prototypical systems should be replaced with actual protocols supported by the Space Station Freedom.

7. APEX Testbed

The proposed testbed for DAA is a real-time monitoring and diagnosis expert system called APEX (Autonomous Power EXpert) for the electrical power distribution system of the Space Station Freedom [23], [24]. We will use APEX to illustrate how DAA can be applied to real-time knowledge-based systems for Space Station Freedom. It was previously implemented in KEE and Common Lisp and is being ported to ART-Ada and Ada at NASA Lewis Research Center. The APEX testbed will be used to demonstrate the advantages of this approach.

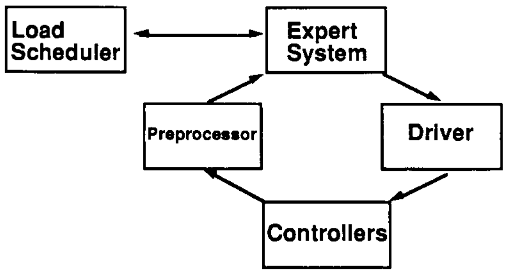


Figure 7-1: Current APEX

Figure 7-1 is a simplified block diagram of the current APEX implementation while Figure 7-2 is that of the new implementation based on DAA. In the current implementation of APEX, there are three modules:

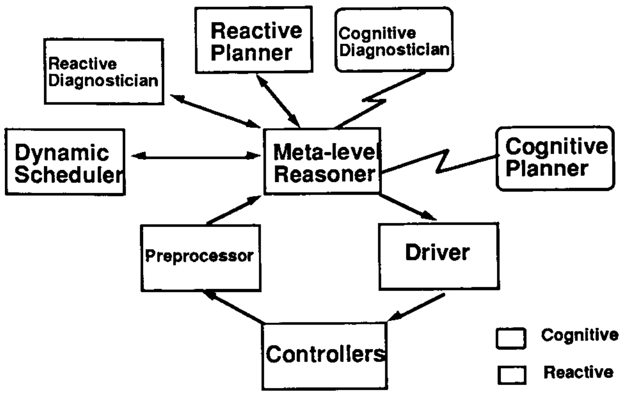


Figure 7-2: APEX based on DAA

- an expert system module written in KEE and Common Lisp that detects multiple faults, predicts possible future faults, and recommends fixes;
- a scheduler module written in C based on linear programming that schedules electrical power distribution for maximum utilization of generated electrical power; and
- several software controller modules written in Ada that detect single faults and fix them immediately [25].

The software controller modules are written in Ada and deployed on the hardware controllers of the electrical power distribution system. These modules are designed to meet timing requirements of less than a second. They are examples of reactive agents.

The scheduler module is implemented separately from the expert system module, and runs on a PC communicating through a network. It is expected to be deployed on the Station as a reactive agent because its absence is unacceptable when the transmission between the Station and the control center is down. This module seems to lack dynamic scheduling capability. We intend to investigate the possibilities of applying AI techniques for dynamic scheduling. NASA Lewis Research Center is also considering COMPASS (COMPuter Aided Scheduling System). COMPASS is an interactive planning and scheduling system developed by McDonnell Douglas, and is available through NASA Johnson Space Center [3]. It is written in Ada and uses X windows interfaces.

The expert system module should be distributed; more critical functionality that requires reactive responses should be separated as a reactive diagnostician and deployed on the Station while less critical functionalities such as trend analysis and long-term prediction can remain as a cognitive diagnostician in the ground-based control center. Following [8], [7], the reactive diagnostician based on *associative* reasoning methods will be implemented as a Bayesian belief network while the cognitive diagnostician based on rule- and model-based reasoning methods will be implemented in ART-Ada. By the same token, a recovery planner may have to be separated into a reactive planner and a cognitive planner. It is our intention to investigate the possibilities of adopting reactive planning methods found in various literatures [1], [5], [10], [17], [18] to implement a reactive planner.

8. Conclusion

DAA focuses on the cooperation between onboard systems and ground-based ones, which is not currently well addressed by the Space Station Freedom Program. It is not easy to achieve cooperative processing between onboard systems and ground systems. We believe that it is technically feasible, but it is difficult because it involves multiple organizations. Currently, onboard systems and ground-based systems are handled by different contractors. If an architecture like DAA is adopted as a general framework for the Space Station, it could be used as a "glue" between different contractors.

Many flight-related software components will reside in the SSCC (Space Station Control Center) because onboard computing resources are very limited. We believe that ground-based flight-related software systems should operate in the same environment as onboard flight software for two reasons:

- If ground-based software components are crucial for flight, it should be considered as part of the flight software. The same verification and validation standard that is normally applied to onboard flight software should also be applied to these software components.
- If ground-based software components are destined to migrate to the Station, it would be essential for the SSCC to have the same operating environment as the onboard environment.

Because of these reasons, the Ada mandate should be imposed on the development of any new ground-based flight-related software components as well as onboard software.

Another important issue raised by DAA is the assessment of risks caused by communication delays. Average communication delay may be less than a minute in normal operating conditions, which is not significant. On the other hand, there might be longer delays caused by "blind spots" in the communication networks or by hardware failures in the transmission systems. NASA should assess any risks of having catastrophic failures on the Station due to the absence of support from ground-based systems during these communication delays.

9. Acknowledgments

The author wishes to acknowledge the guidance and support of Chris Culbert and Bob Savely of NASA Johnson Space Center, Greg Swietek of NASA Headquarters, and Captain Mark Gersh of the U.S. Air Force. Brad Allen, Mark Auburn and Sherry Walden of Inference Corporation contributed to the project. Barbara Hayes-Roth of Stanford University, Rajendra Dodhiawala and Cindy Pickering of FMC, Francois Felix Ingrand of SRI International, Tom Broten of TRW, Rich Knackstedt and Steve Bate of McDonnell Douglas, Jerry Walters of NASA LeRC and many other NASA scientists and contractors provided useful discussions and feedback.

References

1. Agre, P., Chapman, D. Pengi: An Implementation of a Theory of Activity. Proceedings of the International Conference on Artificial Intelligence, AAAI, 1987.

2. Ash, D., Hayes-Roth, B. Temporal Representations in Blackboard Architectures. Tech. Rept. KSL 90-16, Knowledge Systems Laboratory, Stanford University, March, 1990.

3. Bayer, S.E. Space Station Freedom Program Capabilities for the Development and Application of Advanced Automation. Tech. Rept. MTR-89W00279, The MITRE Corporation, December, 1989.

4. Dodhiawala, R. et. al. Real-Time AI Systems: A Definition and An Architecture. Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1989.

5. Drummond, M. Situated Control Rules. Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems, University of Rochester, 1988.

6. Ginsberg, M.L. "Universal Planning: An (Almost) Universally Bad Idea". *AI Magazine 10*, 4 (1989).

7. Hayes-Roth, B. Architectural Foundations for Real-Time Performance in Intelligent Agents. Tech. Rept. KSL 89-63, Knowledge Systems Laboratory, Stanford University, December, 1989.

8. Hayes-Roth, B. et. al. Intelligent Monitoring and Control. Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1989.

9. Ishida, T. Optimizing Rules in Production System Programs. Proceedings of the National Conference on Artificial Intelligence, AAAI, 1988.

10. Kaelbling, L.P. Goals as Parallel Program Specification. Proceedings of the National Conference on Artificial Intelligence, AAAI, 1988.

11. Lee, S.D. Toward the Efficient Implementation of Expert Systems in Ada. Submitted to the TRI-Ada Conference, ACM, 1990.

12. Lee, S.D. A Distributed Agent Architecture for Real-Time Knowledge-Based Systems. Inference Corporation, May, 1990.

13. Lee, S.D., Allen, B.P. Deploying Expert Systems in Ada. Proceedings of the TRI-Ada Conference, ACM, 1989.

14. Lee, S.D., Allen, B.P. ART-Ada Design Project - Phase II, Final Report. Inference Corporation, February, 1990.

15. Lee, S.D., Allen, B.P. ART-Ada: An Ada-Based Expert System Tool. Proceedings of the Space Operations, Applications and Research Symposium (SOAR), NASA, 1990.

16. Lim, M.H., Takefuji, Y. "Implementing Fuzzy Rule-Based Systems on Silicon Chips". *IEEE Expert 5*, 1 (February 1990).

17. Nilsson, N.J. Action Networks. Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems, University of Rochester, 1989.

18. Schoppers, M.J. Universal Plans for Reactive Robots in Unpredictable Domains. Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1987.

19. Schwartz, T.J. "Fuzzy Systems Come to Life in Japan". *IEEE Expert 5*, 1 (February 1990).

20. Sha L., Goodenough J.B. Real-Time Scheduling Theory and Ada. Tech. Rept. CMU/SEI-89-TR-14, Carnegie-Mellon University, Software Engineering Institute, April, 1989.

21. Sha, L., Goodenough, J.B. "Real-Time Scheduling Theory and Ada". *Computer 23*, 4 (April 1990).

22. Togai, M., Watanabe, H. "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning". *IEEE Expert 1*, 3 (Fall 1986).

23. Truong, L., et. al. Autonomous Power Expert Fault Diagnostic System for Space Station Freedom Electrical Power System Testbed. Proceedings of the Workshop on Space Operations Automation and Robotics, NASA Johnson Space Center, July, 1989.

24. Walters, J.L., et. al. Autonomous Power Expert System. Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence, NASA Goddard Space Flight Center, May, 1990.

25. Wright, T, Mackin, M., Gantose, D. Development of Ada Language Control Software for the NASA Power Management and Distribution Testbed. NASA Lewis Research Center, 1989.

26. Zadeh, L.A. "Fuzzy Logic". *Computer 21*, 4 (April 1988).

I. Proposed Real-Time Extensions to ART-Ada

I.1. Performance Monitoring and Tuning

The performance of an expert system varies widely depending on how it is implemented. It is often necessary to monitor activities in the pattern matcher (e.g. the number of pattern instantiations, partial matches, activations, etc.) or the execution time of a rule RHS (right-hand side) action in order to determine areas for optimization. Performance analysis can be aided by a set of tools that graphically display the information.

Unlike conventional software, rule-based systems are sensitive to the ordering of patterns in rules. Currently, the only way to optimize pattern ordering is to monitor activities in the pattern and join networks and optimize them manually. It may be possible, however, to automate this manual optimization process. It has been reported that an automated tool was successfully used to optimize join ordering [9]. An optimization algorithm can be automatically applied to a rule-based program to find near-optimal pattern ordering for the entire program.

I.2. Temporal Reasoning and Trend Analysis

In a real-time expert system, it is often necessary to reason about and perform statistical analysis on *temporal* data -- data that change over time. In order to avoid information overloading, several levels of abstraction should be used. Raw data should be preprocessed to suppress noises and redundant data. Historical data should not participate in the pattern-matching process directly. Rather, high-level abstraction acquired by applying temporal reasoning and trend analysis to the historical data, should used in the knowledge base.

We propose to implement a set of functions that can be layered on top of ART-Ada as a separate library for temporal reasoning and trend analysis. This library is based on the concepts, *monitors*, *events* and *timers*. A *monitor* is used to store historical data in a ring buffer outside of the knowledge base. A monitor is referred to only by its name, which is stored in a hash table. *Events* are used to extract temporal relations between parameters. *Events* are a collection of time that satisfies certain conditions. Rule-based systems are usually data-driven. In a real-time system, however, processing must be driven by time as well as data. A *timer* can be used to implemented time-driven processing. For more details on *monitors*, *events* and *timers*, see [12].

I.3. Dynamic Rule Priority

In real-time AI architectures, the priority of a task should be dynamically determined based on the timing constraints and the resource requirements of the task [8], [4]. In the current version of ART-Ada, the priority of a rule cannot be changed dynamically. If the priority of a rule is allowed to be changed at runtime, rule scheduling strategy can also be modified dynamically.

In the following example, the closer the distance is, the higher priority will be assigned to the rule activation. In fact, the same rule can be activated with different priorities if its priority can be modified dynamically. In order for the rule dynamic priority to function properly, the priorities of all activated rules in the *agenda* must be refreshed before a rule is selected for execution.

If the execution time of a rule is known, it can be used to adjust its priority. It is often desirable to assign a higher priority to a rule with a shorter execution time. In fact, it is the strategy used by the rate monotonic theory [20], [21]. In the following example, duration is the execution time of a rule RHS action. The execution time can be either measured or estimated.

```
(defrule foo
  (declare (salience ?s = 1/?d))
  (declare (duration 1 sec))
  (schema ?enemy-plane (distance ?d))
  =>
  (...))
```

I.4. Message Passing between Distributed Expert Systems

Multiple cooperating ART-Ada applications can run on loosely-coupled multiple processors. ART-Ada supports object-oriented programming. A *method* is a function associated with an object or a class that can be inherited. When a message is sent via an ART-Ada function *send*, an appropriate method will be invoked. If objects are distributed over multiple processors, and a data dictionary is used to define mapping between a processor and an object, the message passing mechanism through *send* can be used without modification to implement distributed message passing. When a message is sent, the system can simply check the data dictionary and send the message to the appropriate processor. Each ART-Ada application can use an asynchronous function to check its message queue between every rule firing.

N91-20689

Autonomous Power Expert System Advanced Development

Todd M. Quinn
Sverdrup Technology, Inc.
NASA Lewis Research Center Group
2001 Aerospace Parkway
Brook Park, Ohio 44142

Jerry L. Walters
National Aeronautics
and Space Administration
21000 Brookpark Road
Cleveland, Ohio 44135

ABSTRACT

The Autonomous Power Expert (APEX) system is being developed at NASA Lewis Research Center to function as a fault diagnosis advisor for a space power distribution test bed. APEX is a rule-based system capable of detecting faults and isolating the probable causes. APEX also has a justification facility to provide natural language explanations about conclusions reached during fault isolation. To help maintain the health of the power distribution system, additional capabilities have been added to APEX. These capabilities will allow detection and isolation of incipient faults and enable the expert system to recommend actions/procedures to correct the suspected fault conditions. New capabilities for incipient fault detection consist of storage and analysis of historical data and new user interface displays. After the cause of a fault has been determined, appropriate recommended actions are selected by rule-based inferencing, which provides corrective/extended test procedures. Color graphics displays and improved mouse-selectable menus have also been added to provide a friendlier user interface.

This paper contains a discussion of APEX in general and a more detailed description of the incipient detection, recommended actions, and user interface developments during the last year.

INTRODUCTION

Our future presence in space will require larger and more sophisticated working and living environments. Such environments will consist of numerous integrated subsystems that will have to be maintained with a high degree of reliability. Primary among the various subsystems is the power distribution system that supplies electrical energy throughout the space-based facility. The availability of space power will be finite, and the sharing of limited power resources will have to be optimally scheduled. If a fault occurs within the power distribution system, disruption of scheduled power usage will result in a costly loss of mission time and could threaten the operation of other subsystems such as life support.

Figure 1 shows a typical power distribution test bed designed for space-based applications. Electrical energy is collected by solar arrays, converted to 20 kHz power, and transmitted through power lines to the various loads. Power distribution paths are opened/closed by using switching devices known as Remote Bus Isolators (RBI's). Each RBI contains a number of sensors to measure the various operating parameters of the power distribution system such as current, voltage, power, and power factor. Upper level controllers access the sensory data and relay the information to a central Power Management Controller (PMC). When an RBI is tripped because of an overcurrent condition attributed to a fault in the system, the PMC will attempt to restore the lost power by activating alternate RBI's that will reconfigure the power distribution system.

Quick and automatic reconfiguration of the power distribution system by the PMC provides the necessary capability to maintain power distribution when a fault occurs. To preserve the health of the power distribution system, however, the fault must be isolated and appropriate recovery procedures must be performed to repair the problem. Potential power disruptions can also be avoided by detecting incipient fault conditions that are, at present, nonthreatening to the power distribution system but that over a period of time will become a fault. Isolation of and recovery from a fault condition depend on the technical knowledge and experience of power systems personnel. Incipient faults are detected by continuously monitoring sensory data for indications of persistent upward or downward trends in any of the power distribution system measurements.

In a real space environment, with a limited crew size, space power expertise may be unavailable, and with a large number of switching devices, routine maintenance checks and power system data analyses for incipient fault conditions would require a significant amount of crew time. Therefore autonomous control of space power distribution by expert systems with fault isolation, fault recovery and incipient detection will greatly enhance the reliability of the power distribution system and reduce the human workload.

The Autonomous Power Expert (APEX) is a software system designed to emulate a human expert's reasoning processes in order to solve problems in space power distribution. The APEX system automatically monitors the operating status of the power distribution system and reports any anomaly as a fault condition. APEX then functions as a diagnostic advisor, aiding the user in isolating the cause of the detected fault condition and in repairing the power distribution system.

Development work for the current design of APEX was based on the Power Distribution Unit A (PDUA) subsystem shown in figure 1 [Troung 1989]. APEX is currently interfaced to the PMC controller, which communicates with the Power Distribution Controller (PDC). APEX sends a request for data to the PMC. The PMC acquires the requested data from sensors on the power distribution switching devices via the PDC and passes the data to APEX. When APEX has collected the power distribution parameter data, a fault detection phase is initiated.

APEX detects faults by comparing expected values to the measured operating values (parametric values) obtained from the controller. The expected values are calculated by APEX from the scheduled profile data of the loads connected to the PDUA. If no deviations from the expected operating state of the PDUA are found, APEX will again request data from the PMC and re-initiate the fault detection activity with the new data. If an anomaly is found within the data acquired from the PMC, APEX will inform the user that a fault has been detected.

The user can direct APEX to isolate the probable cause of the fault. APEX accesses information and rules contained in its knowledge base, reaches a conclusion, and displays to the user the probable cause for the detected fault. The user can then ask APEX to justify its conclusion and to recommend actions to correct the fault.

IMPLEMENTATION OVERVIEW

APEX is currently implemented on a Texas Instruments Explorer II workstation in LISP and employs the Knowledge Engineering Environment (KEE) expert system shell. APEX consists of an integrated set of software, including a knowledge base, a database, an inference engine, and various support and interface software. The knowledge base comprises facts and rules that correspond to knowledge acquired from the human expert during problem solving. The database is the basic working area where storage and calculations of sensory data for incipient fault detection occurs. The inference engine is the reasoning mechanism that, during fault isolation, draws conclusions from information stored within the knowledge base. In choosing the appropriate recovery procedures for the isolated fault, APEX also relies on the reasoning capabilities of the inference engine. Conventional software provides the user with an interactive interface to communicate with APEX and to obtain data from various sources such as power distribution hardware and planner/scheduler software.

Knowledge is represented within the APEX knowledge base mainly by frames, semantic triples, and production rules. Frames are structures that describe objects or classes of objects and their relationships. Objects are composed of slots that specify the various attributes belonging to each object. Individual slots of an object can contain declarative information or attached procedural functions. Declarative information expresses facts about the object, whereas procedural functions are programs or a set of procedural steps attached to the slot producing a particular behavior for the object. Within APEX, declarative information is represented by semantic triples that state information in the form of object/attribute/value (ie. attribute of object = value). Production rules are "If-Then" statements that imply either declarative facts or procedural behaviors when the conditional statements contained in the premises of the rule are found to be true. [Sell 1985]

The database contains a historical record of data acquired from the switching devices in the power distribution system. Storage and manipulation of these data are accomplished with conventional techniques and do not require the use of the inference engine. A detailed description of the structure and use of the database is given in the section on incipient fault detection.

APEX employs an inference engine contained in the Knowledge Engineering Environment (KEE) expert system shell [KEE 1989]. The inference engine is the heart of the expert system; it determines how knowledge is represented and processed. By operating on the rules within the knowledge base, the inference engine can reason and draw inferences about the state of the power distribution system. The inference engine rule processing strategies are commonly referred to as forward and backward chaining. Forward chaining works from the given data to a conclusion by examining the premises of the rules to determine if the conclusion of a rule can be inferred. If a conclusion is inferred, the new facts asserted by the conclusion could then cause other premises in other rules to imply even more conclusions. Backward chaining works from a particular goal and tries to either confirm or refute its truth. In the case of backward chaining, rules are selected by first matching the conclusion of the rules with the stated goal. If the true/false values of the premises of the matched rule are unknown, the premises become subgoals, which then can cause other rules to be selected. The goal is asserted only when all of the premises and subgoals of the goal-matched rule are known to be true. In the APEX system, fault detection is driven by sensory data and is implemented with forward chaining. Fault isolation is accomplished with backward chaining by giving APEX the goal of finding the probable cause of the fault.

APEX also consists of various support software that allows communication with the outside world. The user interface enables APEX to communicate with the operator through color graphics display screens and menu selections. Using the menu options, the user can select the detail level of information to be displayed, ask for justification of a particular conclusion, and request recommended action to correct an isolated fault. Other communication links provide data acquisition from the power distribution system via the lower level controllers, and load profile data acquisition from a remote scheduling system [Ringer 1990].

Incipient Fault Detection

Faults are detected by comparing the parametric values (measured operating values) of the power distribution system to the expected values and identifying any abnormal operating parameters. When the detection rules have been exhausted, APEX reports to the user whether or not any faults were detected. If a fault was detected, the user can then ask the expert system to isolate the probable cause of the fault. If no abnormal conditions were detected, the historical data is analyzed for incipient fault conditions.

Incipient detection is based on

statistical linear regression and correlation analysis of the historical data. As new data are received, the parametric values of the power distribution system are stored as historical data under the appropriate attributes for each switching device. Along with each measured value, the expected value that is calculated by the expert system is also saved. The expert system analyzes the historical data looking for any indication of a parametric attribute that has maintained either an upward or downward trend in the data values over a period of time. The following parametric attributes are stored for each device: switch A current, switch B current, line voltage, load voltage, and power.

Since the power system is dynamic and the measured value fluctuates over a period of time during normal operation, a parametric ratio of the measured-to-expected value is used to identify any increasing or a decreasing trends in the parametric data. Thus, if the measured and the expected values are equal, the ratio will be one. If the measured value is higher than the expected value, the ratio will be greater than one; if the measured value is less than the expected value the ratio will be less than one.

Once the data have been stored in the database, correlation coefficients are calculated for each parametric attribute of each switching device. The correlation coefficients are calculated in the following manner [Trivedi 1982]:

The mean value of a variable is found from

$$\bar{a} = \frac{1}{N}\sum_{1}^{N} a_n$$

the time variance from

$$\sigma_x^2 = \overline{X^2} - (\bar{X})^2$$

the parametric variance from

$$\sigma_y^2 = \overline{Y^2} - (\bar{Y})^2$$

and the covariance of X and Y from

$$\overline{XY} - \bar{X}\bar{Y}$$

where X is the time values and Y is the parametric values.

The correlation coefficient r, then, is

$$r = \frac{\overline{XY} - \bar{X}\bar{Y}}{\sigma_x \sigma_y}$$

where the standard error is

$$S_y = \sigma_y \sqrt{1 - r^2}$$

the slope is

$$m = \frac{\overline{XY} - \bar{X}\bar{Y}}{\sigma_x^2}$$

and the Y-intercept is

$$b = \bar{Y} - m\bar{X}$$

A high correlation coefficient, caused by a parametric ratio trend, indicates that a temporal relationship exists. The value of the correlation coefficient lies between zero and one. A zero indicates that there is no correlation between the time and historical parametric data; however the closer correlation coefficient is to one, the stronger the time and parametric value correlation. APEX currently will consider an incipient fault condition to exist if the correlation coefficient of a parametric attribute is higher than .75.

Once an incipient fault condition has been detected, the user can view the results of the statistical analysis and also have APEX isolate the probable cause of the incipient condition. Figure 2 shows a typical display indicating a definite increasing trend in the ratio between measured values and expected values. The trend was detected within the switch A current parameter of switching device RBI.3/3. Along with the plot of the linear regression results, the correlation coefficient, slope, standard error, and y-intercept are displayed for the user. A set of isolation rules for detected incipient fault conditions can access the database and examine correlation coefficients of the various parametric attributes of each switching device.

USER INTERFACE

The goal of the user interface is to provide access to APEX which is intuitive, and requires only a small amount of training. Communication between APEX and the user is accomplished with easy to use mouse-selectable menus, and color graphics and text displays. The user interface screen presents a color display that is divided into three areas as shown in figure 3. The top portion of the screen is the control menu that allows the user to select the desired APEX function. When a function is selected, mouse-selectable options for that function appear in the options menu located in the lower portion of the screen. As APEX performs the selected function, the control menu is replaced by a status display window indicating the operational steps being executed. Fault detection and fault isolation results are shown within the main display area by means of color diagrams and text explanations.

The control menu contains the following six mouse-selectable functions: MONITOR, DETECTION, ISOLATE CAUSE, RESET SYSTEM, LOG FILE, and EXIT. The MONITOR selection causes APEX to continuously acquire and check parametric values from the power distribution system. When either an active or incipient fault is detected, APEX stops monitoring and displays a "fault detected" message in the upper left corner of the user interface screen. Once alerted, the user can display the fault detection analysis performed by the MONITOR function by selecting DETECTION in the control menu. When ISOLATE CAUSE is selected from the menu, APEX will access the fault isolation rules to determine the probable cause of the detected fault. The RESET SYSTEM function clears the working space of the APEX system to prepare APEX for monitoring the power distribution system. If the user wants to record the session with APEX, a file can be opened/closed and printed with the LOG FILE function. The EXIT function allows the user to either terminate APEX, switch over to the power system data simulator, or to communicate with a remote planner/scheduler.

Recall that when a function is selected, the options menu provides the user with available options for that function. For example, when the user selects the ISOLATE CAUSE function, APEX will display the probable cause of a detected fault and the options menu will contain CONTINUE, WHY?, RECOMMEND. The CONTINUE option will allow the user to exit from the ISOLATE CAUSE function and continue APEX operations with the control menu. If the user selects WHY?, APEX will display the reasoning process leading to the probable cause conclusion. The RECOMMEND option allows the user to request recommended action procedures for correcting the fault; this option also has a user confirmation/rejection sub-option during any procedural step requiring autonomous action of the APEX system, such as reconfiguring the power distribution system.

The graphical displays in the main display area consist of a set of hierarchical diagrams that represent three different levels of information. The diagram in the main display area shown in figure 3 represents the overall power distribution system. When an active fault is detected, in the diagram the area of detection is outlined in red and a red flashing cursor appears next to the area. For an incipient fault condition, the area is outlined in yellow and has a yellow flashing cursor. The yellow indicates that a parametric value is probably going to go out of tolerance if preventive action is not taken. The user can get a more detailed diagram of an area by choosing the particular area of interest and clicking the mouse. Figure 4 shows the user interface screen after the user selects on PDUA of the top level diagram. In this PDUA subsystem diagram, the user can easily see the location of the detected parametric abnormality at the switching device level. Figure 5 shows the switch level diagram after the user clicks the mouse on one of the switching devices, such as RBI 3/3. Each switch level diagram displays the actual measured data values enabling the user to see which parametric attribute is out of tolerance.

RECOMMENDED ACTIONS

After APEX has isolated the probable

cause of a detected fault or an incipient fault condition, the user can to ask for fault recovery recommendations. APEX will analyze available information about the current operating conditions with respect to the fault and display appropriate actions to be taken. Recommended actions pertain to both short- and long-term recovery. Short-term recovery determines if the fault can be tolerated for a period of time, if the power distribution can be reconfigured, or if load shedding is necessary. For long term recovery, the repair procedures needed to correct the fault are determined after short term actions have been implemented.

Short-term recovery analysis is based on a set of "recommended action" rules for the particular fault condition. Information about available power sources, current configuration of the power distribution system, the scheduled run times of the loads, and the effects of the fault on the system are all considered during the analysis. If enough power is available and the effects of the fault are minimal with respect to remaining scheduled run time of the affected loads, then the fault can be tolerated and the loads are allowed to run to completion. If the fault is seriously affecting the amount of power reaching a particular load and an alternate path for power distribution exists, then the system can be reconfigured automatically, or with user confirmation, to allow the load to run to completion. When the fault cannot be tolerated and alternate power distribution paths are unavailable, then the schedule for the loads is replanned by a remote scheduling agent; this results in load shedding and a new schedule.

After short-term recovery, the fault in the power distribution system needs to be repaired. The appropriate procedures needed to repair the power distribution system are determined by long term recovery, which is also based on a set of recommended action rules. In some cases, the cause of the fault is localized to a group of possibilities, and additional troubleshooting procedures are displayed to intelligently guide the user to further isolate the exact location and to make repairs.

CONCLUDING REMARKS

The APEX system consists of an integrated set of software agents, including a knowledge base, database, inference engine, data acquisition interface to the power distribution hardware, and a communication interface to a remote planner/scheduler. During the past year, advanced development of the APEX system has included addition of incipient fault analysis, an improved multilevel color user interface and a new recommended action facility.

Incipient fault analysis adds a unique health monitoring capability to prevent faults by continuously monitoring all parametric values in the power distribution system. APEX can warn the user of potentially threatening fault conditions before power distribution interruptions are experienced. This continuous health monitoring will of relieve human operators of labor-intensive mission control operations. Moreover, the type of continuous monitoring that APEX provides eliminates problems that can occur with human monitoring such as errors caused by fatigue.

The color capability of the new user interface enhances the information display and provides a friendlier man machine interface. Location and type of detected faults are immediately recognized when flashing combined with color coding appears on multilevel displays. In addition, the user interface contains mouse-selectable menus that present appropriate options for accessing information and obtaining fault recovery/prevention assistance.

The new recommended actions feature determines the most appropriate procedures for recovering from and preventing power distribution faults. The procedures are determined by rules stored in the knowledge base and the reasoning capability of APEX. Recommended actions consist of both short- and long-term recovery procedures necessary for maintaining the health of the power system. Execution of short-term recovery procedures restores power to scheduled loads, and execution of long-term actions effectively repairs isolated areas of the power distribution circuit.

In future space applications, APEX can be applied to help maintain the operational health of the power distribution systems.

APEX will be able to diagnose fault conditions and recommend appropriate recovery procedures when experienced power system personnel are unavailable. By allowing APEX to autonomously monitor and analyze power distribution system data, faults can be detected before serious problems develop and costly power interruptions occur. Increased reliability of space power distribution and a substantial reduction in the human labor required for routine monitoring of system operations is the goal of the APEX project.

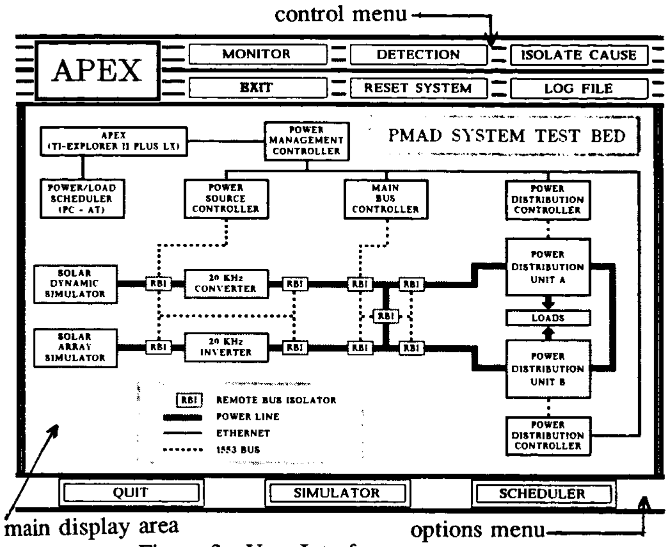


Figure 3. User Interface (with power system diagram)

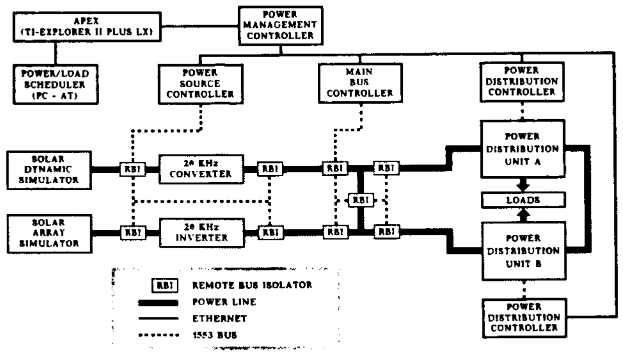

APEX (TI-EXPLORER II PLUS LX)
POWER MANAGEMENT CONTROLLER
POWER/LOAD SCHEDULER (PC - AT)
POWER SOURCE CONTROLLER
MAIN BUS CONTROLLER
POWER DISTRIBUTION CONTROLLER
SOLAR DYNAMIC SIMULATOR
20 KHz CONVERTER
SOLAR ARRAY SIMULATOR
20 KHz INVERTER
RBI
POWER DISTRIBUTION UNIT A
LOADS
POWER DISTRIBUTION UNIT B
POWER DISTRIBUTION CONTROLLER
RBI REMOTE BUS ISOLATOR
POWER LINE
ETHERNET
1553 BUS

Figure 1. Power Distribution Test Bed

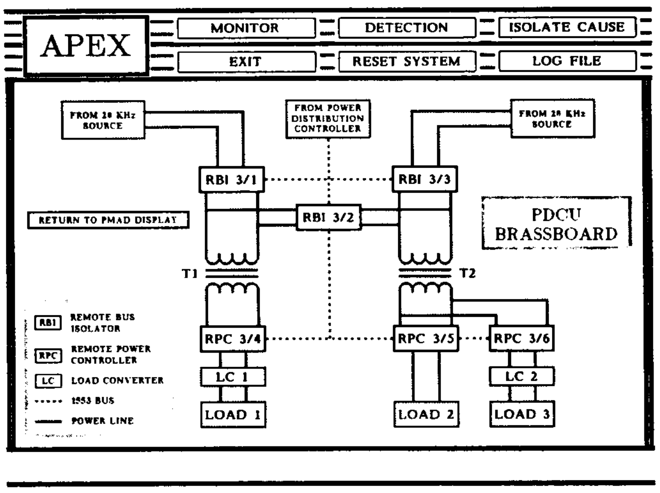
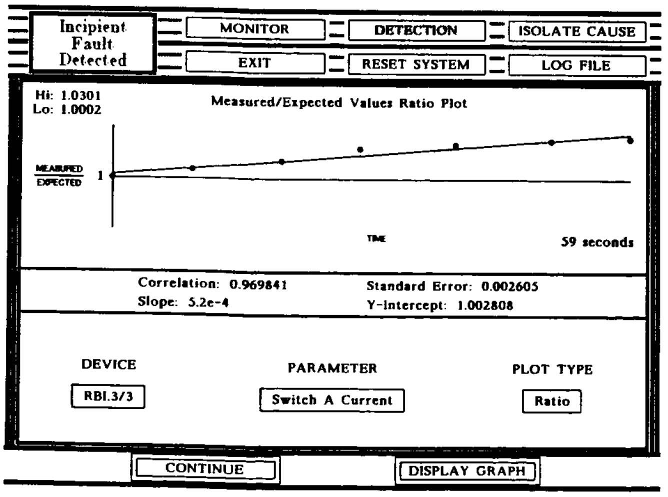


Figure 4. User Interface (with PDUA diagram)



Hi: 1.0301
Lo: 1.0002

Measured/Expected Values Ratio Plot

59 seconds

Correlation: 0.969841 Standard Error: 0.002605
Slope: 5.2e-4 Y-Intercept: 1.002808

DEVICE: RBI.3/3
PARAMETER: Switch A Current
PLOT TYPE: Ratio

Figure 2. Incipient Fault Condition Analysis

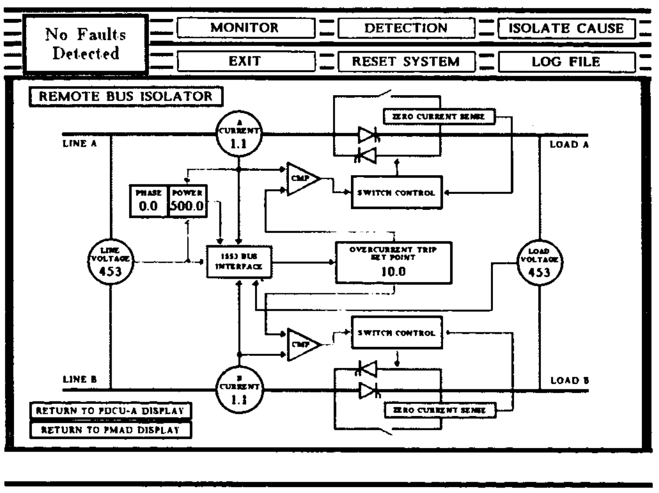


Figure 5. User Interface (with switch diagram)

ACKNOWLEDGMENTS

We would like to acknowledge the following employees of Sverdrup Technology Inc., Lewis Research Center Group: Walter Krawczonek, our domain expert, for providing the knowledge used in developing the recommended actions feature of APEX; Anthony Merolla for technical hardware support; and Mark Ringer for his scheduler/planner work. We are also grateful to James Kish and Edward Petrik of NASA Lewis for project leadership and Anoosh Shahidi from Sverdrup for his review of this paper.

REFERENCES:

[Truong 1989] Truong, L., et al.: Autonomous Power Expert Fault Diagnostic System for Space Station Freedom Electrical Power System Testbed, Third Annual Workshop on Space Operations Automation and Robotics (SOAR 1989), (NASA CP-3059).

[Sell 1985] Sell, Peter S.: Expert Systems - A Practical Introduction, Halsted Press, 1985.

[KEE 1989] KEE User's Guide, Version 3.1, Intellicorp, Inc., Mountain View, Ca, May 1989.

[Ringer 1990] Ringer, M. J., et al.: "Autonomous Power Expert System", to be presented at the 25th Intersociety Energy Conversion Engineering Conference, Reno, NV, AIChE 1990.

[Trivedi 1982] Trivedi, K. S.: Regression, Correlation, and Analysis of Variance, Probability & Statistics With Reliability, Queuing and Computer Science Applications, K. S. Trivedi, ed., Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982.

KNOWLEDGE ACQUISITION DESIGN ENVIRONMENT

N91-20690

A Knowledge-Based approach to Configuration Layout, Justification and Documentation

F. G. Craig, D. E. Cutts, & T. R. Fennel
Boeing Computer Services
M/S JA-74
Huntsville Artificial Intelligence Center

C. M. Case & J. R. Palmer
Boeing Aerospace & Electronics,
Huntsville Division
M/S JY-33
P.O. Box 240002
Huntsville, AL 35824-6402

Abstract

This paper describes the design, development, and implementation of a prototype expert system which could aid designers and system engineers in the placement of racks aboard modules on Space Station Freedom. This type of problem is relevant to any program with multiple constraints and requirements demanding solutions which minimize usage of limited resources. This process is generally performed by a single, highly experienced engineer who integrates all the diverse mission requirements and limitations, and develops an overall technical solution which meets program and system requirements with minimal cost, weight, volume, power, etc. This "systems architect" performs an intellectual integration process in which the underlying design rationale is often not fully documented. This is a situation which lends itself to an expert system solution for enhanced consistency, thoroughness, documentation, and change assessment capabilities.

1.0 General Configuration Definition Issues

One of the major issues faced by any aerospace program is the need to consistently apply requirements, constraints, and resources to optimize the layout of equipment in an end item deliverable piece of hardware in the midst of changing environments. The change mandates can result from changing customer requirements, newly derived requirements, reduced program budgets, technological influences or personnel changes. All these changes tend to impact engineering processes, often rendering current approaches inappropriate or current solutions inadequate. In the remainder of this section we present a list (by no means exhaustive) of general issues which must be faced throughout a program's life cycle :

• Fleeting expertise : Turnover of domain experts represents a serious drain on program continuity and often causes work to be adversely impacted since significant portions of domain knowledge and program history often reside with individuals.

• Productive use of resources : Much layout work is both repetitive and resource intensive in nature. Allowing for automation of such repetitive tasks to be accomplished early in the process results in more resources being available for "real" engineering work to be performed later. This is usually a direct result of complimenting engineering expertise with tools which allow problems to be solved at a more abstract level and to off-load the repetitive portions of the task to the automated process. For example, engineering resources may be diverted to cost proposed changes, document accepted changes, and implement new procedures. Because of this type of required reaction, program continuity and productivity can be affected.

• Documentation of engineering rationale : All major programs have periodic requirements to review progress and to answer not only the question of "What has been done?", but also "Why was it done this way?", and "Why can it not be done this way?". The last two questions require the documentation, presentation and defense of engineering rationale. The problem is to provide a sound defense in areas where adequate documentation is generally missing, the expertise may have been lost, or rules may not have been codified, consistently applied, or documented.

• Multi-discipline inputs : Decisions made during the configuration process typically originate across several disciplines and organizational boundaries. Disciplines may or may not be aware of the impact of their decisions on other disciplines. These multi-disciplinary inputs to the engineering process highlight the need for a uniform approach to acquiring and representing those inputs.

• Explicit decision parameters and criteria : For engineering problems of any significant complexity, there is a need for the consistent application of clearly defined problem parameters and dynamic criteria to the engineering process. This is particularly true when these parameters and criteria come from various disciplines.

• Limited alternatives : Often the iterative engineering process is not fully utilized beyond a baseline "satisficing" solution (where the result is not optimal but merely satisfies most of the criteria). Little time is left to consider alternative analyses, configurations, or development paths. Better options may be overlooked because no tool/capability exists for quickly modelling and analyzing engineering alternatives.

• Problem of scale : Unfortunately, major program setbacks often occur because engineering solutions which worked well for small problems (or subsets of the larger problem) do not scale up well. This is particularly true when manual engineering approaches which were controllable and acceptable for the smaller problem are applied to large integrated programs.

As mentioned, the above list is not intended to be exhaustive, but is presented to serve as a reference for the next section.

2.0 Rack Layout Problem Description

As an initial test problem we have selected the Space Station Freedom module configuration task. This effort is similar to that required in many aerospace configuration layout applications in terms of complexity, constraints, and resources. It is above average in the number of expected major changes and long period of implementation. These factors make the configuration problem an ideal candidate for a knowledge based system.

The particular test domain area is that of rack placement aboard station modules. The racks provide the physical packaging for station services and functions. The objective of the rack placement process is to position a group of racks aboard modules in a configuration that minimizes utilization of resources, optimizes operational efficiency, and meets as many requirements and constraints as possible. The rack layout problem is representative of various configuration layout problems faced within many aerospace programs. Currently three other potential applications for this type of system have been identified within Work Package 1 of the SSFP, and it is expected that a number of additional spinoff applications will surface. Also, work performed on this project could be applied to areas external to the SSFP (other suggested areas include the outfitting of Commercial Aircraft and the Manned Mars Mission).

We are currently researching knowledge-based systems approaches to aid in this problem. The purpose of the research is to attempt to overcome the following perceived problems.

<u>Fleeting Expertise</u> : Currently, only one person in the Space Station program is identified as an "expert" on rack placement.

Claim : Development of an expert system to document the analysis, criteria, and engineering processes used by the expert will allow knowledge needed to solve the problem to be preserved and to be available for review by "non-experts".

<u>Productive use of resources</u> : The current manual approach to this process is quite time consuming and labor intensive. Rack layout reconfiguration for the station must be performed in step with other changes to the program. Due to lack of time, changes to rack configuration often represents an "acceptable" rather than an "optimal" solution.

Claim : The expert system is expected to significantly reduce the amount of time required to produce a new configuration. Additionally, the rules and procedures used by the system will be applied consistently through the automated program. Also, the expert system doesn't "forget" the rules or procedures during periods when the expert is busy with other task. Indeed, such an expert system could be used to train less skilled personnel to perform the task and can be used by the expert to explain the required analyses and procedures for the rack placement process.

<u>Defense of engineering rationale</u> : SSFP has a requirement for periodic reviews where design decisions must be justified.

Claim: In a rule-based system, the engineering rationale for a particular configuration is implicit in the set of rules used to generate that configuration. This engineering rationale provides placement justification and explanation. One of the objectives of the current work is to extract intelligible rationale from the set of rules used to generate the configuration.

<u>Multi-discipline inputs</u>: A large number of constraints exist between racks within and across modules. When these constraints are imposed on a large number of racks, a difficult constraint problem emerges. This problem is compounded by the fact that these constraints are imposed by different domain areas (such as power, thermal, cost, safety, etc.) and may be physical, functional, or operational in nature.

Claim : Experts from all applicable domain areas provide input to the rules and procedures used for the automated placement process. An additional advantage is that a unified approach to the acquisition, analysis, and representation of this domain knowledge can be developed and more easily verified by the experts from the various disciplines.

<u>Explicit decision parameters & criteria</u> : The lack of a uniform approach to explicitly identify applicable parameters and then consistently apply domain rules for rack placement hinders both the ability to quickly produce optimal rack layouts, and the ability to provide justification for a particular configuration.

Claim : The objects, rules, and associated parameters can be printed, queried interactively, and dynamically changed. This makes explicit the answers to questions such as:

What impact did the rule ....

"IF
the rack is rated as 'noisy' ,
THEN
don't place it near the crew sleeping quarters."

... have on the decision to place the rack?

<u>Limited alternatives</u> : Currently, analysis of rack configurations takes anywhere from several hours for the simplest changes to several weeks for more common changes. As expected, this does not leave much time for analyzing "What if...? situations."

Claim : Once the applicable parameters have been identified, and domain expertise has been captured, this expert system would support the ability to "tweak" priorities and constraints allowing engineers to analyze alternative configurations. The comparative "goodness" of rack configurations could be determined, and the tool could be used to suggest or support engineering change requests. Obviously, this does not imply that human expertise would no longer be needed. Rather it implies that more engineering analysis could be performed and human intuition could be used to fuller advantage by allowing the engineer to work at a higher level of abstraction.

<u>Problem of scale</u> : Currently the Phase 1 SSFP calls for only two modules and 4 nodes on the American portion of the program. Even this first phase of the program requires over 144 racks which must be assigned within a full range of physical, functional, and operational constraints. The multiplicity of constraints and the number of racks makes a manual approach to solving the problem nearly intractable.

Claim : While the number of racks and other "real world" objects is expected to remain relatively constant, it is anticipated that the number of constraint and control rules in the knowledge base will expand. No reliable data was available to estimate the bounds on the number of these rules. The tool selected for implementation set no upper bound on the size of the knowledge base (other than memory limitations). Speed was not a primary issue in this application, but reasonable response time was expected. The expert system incorporates domain expertise to control the focussing of rules which helps to limit the solution search (see the control layer in figure 1). Additional constraints can be easily added (or deleted) as knowledge about the racks and their interactions increases.

3.0 Implementation

The Nexpert expert system building tool from Neuron Data was selected for this project because it offered a number of desired features. Nexpert is a hybrid system supporting the representation of knowledge in objects and rules. It supports full inheritance and procedural attachment of methods as well as forward and backward rule chaining capabilities. It interfaces to user developed external routines as well as PC databases and spreadsheets. In addition, links to large databases such as Oracle and Informix are supported. Within this project we are currently a beta test site for a Hypercard/Nexpert "bridge" which allows communication between Nexpert and Hypercard facilities on the Macintosh II platform. Much of the explanation and training research is currently being performed using Hypercard. Nexpert is C based, runs on a wide range of hardware platforms, and offers a number of relatively inexpensive delivery options.

The prototype system software was implemented using a layered architecture to represent system knowledge (see Figure 1). This layered architecture separates different types of knowledge and aids in development, debugging and maintenance of the system. Chandrasekaran [Ref 3] proposes a similar architecture in which tools might be developed for "problem classes" such as diagnosis or design. He proposes that particular problems within these "problem classes" share similarities and that "generic" approaches to solving them might be appropriate.

Data resides in the lowest layer. The data is currently stored in a spreadsheet format, but facilities exist in the Nexpert tool to retrieve data from a number of sources including PC spreadsheets, PC databases, Oracle and Informix. Data is used to support the next layer representing objects and their associated attributes. These two bottom layers together might be thought of as Object-Attribute-Value (O-A-V) triplets. "Real world" entities such as modules, racks, standoffs, utilities, etc., are represented as objects in the system. Most of this type of knowledge was obtained directly from SSFP documentation. It should also be noted that this data might be obtained during the inference process or from some external source. This external source might well be an external routine which calculates a value and returns it to the object. This is analogous to attaching a "method" to as object.

The constraint layer contains all the constraint knowledge about the particular domain under consideration. This constraint knowledge is stored in rules and is a relatively "flat" knowledge base since this type of knowledge is concerned primarily with only a few "focal" objects (see Figure 2). This knowledge base consists of a collection of

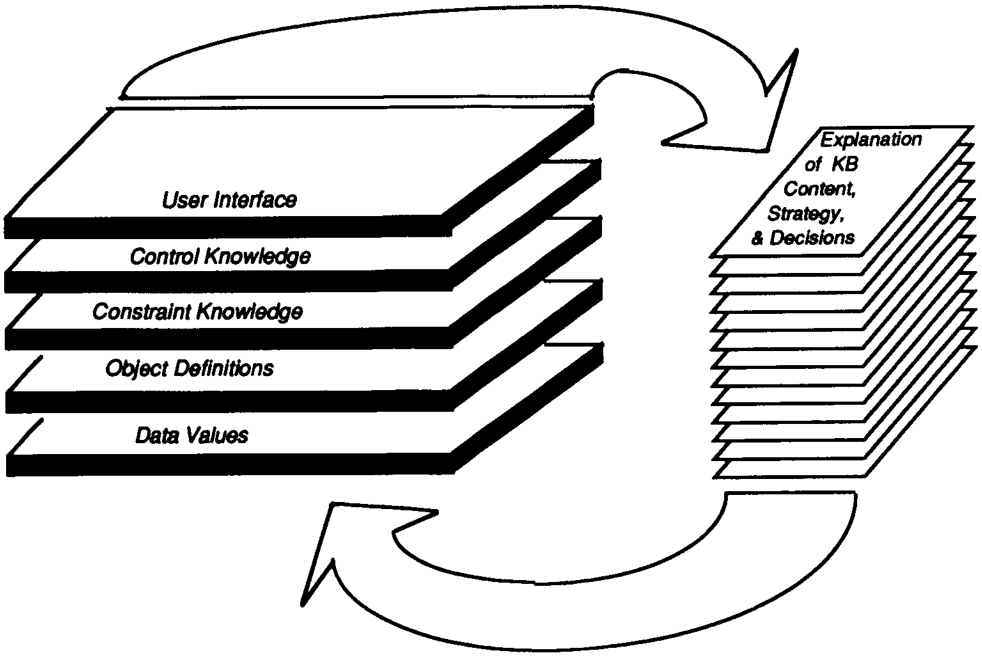


Figure 1 : The Expert System Architecture

ORIGINAL PAGE IS OF POOR QUALITY

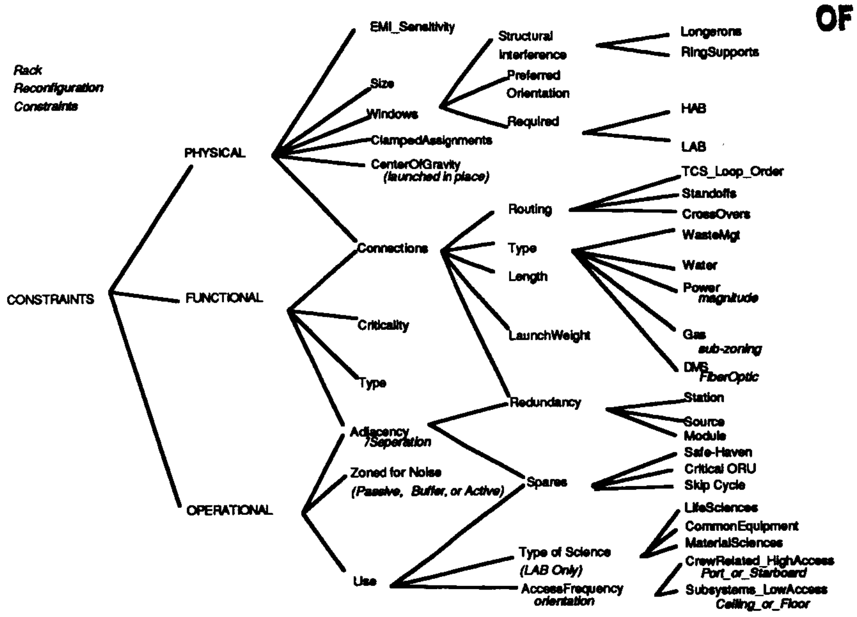


Figure 2 : Constraint rule hierarchy

"microscopic" rules to be used in solving the problem, and do not embody higher level "control knowledge" which a human expert would follow in applying the constraints.

The control knowledge (or meta knowledge) at the next level controls the direction of focus for the constraint knowledge. This level "prunes" the search space so that inappropriate constraint knowledge is not considered. Control knowledge is used to apply the constraint knowledge in much the same way a human expert would. An interesting offshoot of this project has been that the codification of this type of knowledge often helps to better define the problem solving process. This layer is also important in the explanation/justification of design decisions since explanations of design decisions made by the system need to be conveyed in much the same manner as a human expert's explanation.

The user interface represents the user's view of the system. For this application we have designed a "point and click" user interface in which the user manipulates racks within a module configuration. This interface provides input to the control layer about rack(s) to be moved. The control layer applies appropriate constraint knowledge at the next level.

The constraint layer, in turn, obtains needed information from the lower levels. The final result (**no constraints violated, "soft" constraints violated or "hard" constraints violated**) is passed to the user interface where the user can query the system about the particular decision and what support was used in making the decision.

The explanation/justification layer supports access to all the lower levels of the architecture. Queries can be made of objects, constraint knowledge or control knowledge. Explanations differ in content for these different layers and in level of detail based on the level of expertise of the user. A more detailed discussion of this layer can be found in [Ref 1].

4.0 Issues

As with any expert system project, there were a number of issues critical to success. Some of these issues are described below:

Expert availability: From the project's inception, a "domain expert" was identified and has been available at every step in the development process. This expert understands the problem to be solved and is able to articulate his method(s) for solving the problem.

Knowledge Acquisition: The data required for this project comes from both SSFP documents and domain experts. Traditional interview techniques with the primary rack placement expert as well as other experts in related fields have been very successful. Domain experts have recognized the potential utility of such an expert system and have supported it fully. In addition to interviews, the domain experts submitted "test cases" for the system to solve along with their conclusions/justifications as to why the move was good or bad (or could/could not be made). These test cases helped identify many weak areas in the system and helped to build the explanation facilities [Ref 1]. Early in the development process we began to use the system itself to acquire domain knowledge by running test cases against the system. This approach uncovered weak areas in the captured domain knowledge or incorrect assumptions. Thus, the tool itself has been used extensively in the acquisition process.

Verification and Validation: Test cases supplied by the domain experts as well as "working" interviews in which the system is used to test particular rack configurations have been used to test the system for correctness as well as its ability to provide meaningful decision justification. No work has been done to perform tests on the knowledge base for rule subsumption, rule contradiction, or cycles. Much of the system testing will continue to be empirical in nature.

5.0 Future work

One result of both the data acquisition and validation activities was that a larger number of people became aware of the project and began to look for ways to apply the work performed in the project to particular problems in their domain. As a result we anticipate that a number of spinoff projects will emanate from this IR&D work. Problems similar to the rack placement problem include resource allocation problems in which rack resource requirements are matched with resources supplied in the module to maximize the resource utilization. Another similar problem deals with the placement of payload racks (experiments, etc) within the lab module. This task must be performed repeatedly since experiments will continually be moved in and out. Another spinoff of this work may be in the training area. Much of the work being done to provide intelligent design justification and explanation could carry over into training new engineers on the SSF program.

As a component of a training system as well as design justification, we plan to interface this system with simulation systems to provide "deeper" justification or explanation by allowing the user to perform a simulation of a particular configuration during the design process. Adeli [Ref 2] reports on efforts to couple AI techniques with traditional mathematical techniques to aid in the engineering process.

6.0 Summary

Systems incorporating AI technologies to aid design will enhance the engineering process and will ensure that the decisions (and rationale behind them) are available in an intelligible format for future applications. Research in this area and prototype systems such as the one described here will help to clarify and define the engineering design knowledge capture requirements.

References:

[1] Fennel, T.R., et.al, "Graphical Explanation in an Expert System for Space Station Freedom Rack Integration", 5th Conference on Artificial Intelligence for Space Applications, Huntsville, AL, May 1990

[2] Adeli, H. & Balasubramanyam, K.V., "A Novel Approach to Expert Systems for Design of Large Structures", AI Magazine, Winter 1988, pp. 54-63

[3] Chandrasekaran, B. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design", IEEE Expert, Fall 1986, pp. 23-30

N91-20691

CAPTURING, USING, AND MANAGING QUALITY ASSURANCE KNOWLEDGE FOR SHUTTLE POST-MECO FLIGHT DESIGN

H. L. Peters, L. R. Fussell, M. A. Goodwin
Rockwell Space Operations Company
600 Gemini
Houston, Texas 77058-2777
(713)282-2861

R. D. Schultz
Abacus Programming Corporation
14545 Victory Boulevard, Suite 300
Van Nuys, California 91411
(818)785-8000

ABSTRACT

Ascent initialization (I-load) values used by the Shuttle's onboard computers for nominal and abort mission scenarios are verified by a six degree-of-freedom computer simulation. The procedure that the Ascent Post Main Engine Cutoff (Post-MECO) group uses to perform quality assurance (QA) of the simulation is time consuming. Also, the QA information, checklists and associated rationale, though known by the group members, is not sufficiently documented, hindering transfer of knowledge and problem resolution. A new QA procedure which retains the current high level of integrity while reducing the time required to perform QA is needed to support the increasing Shuttle flight rate. Documenting the knowledge is also needed to increase its availability for training and problem resolution.

To meet these needs, a knowledge capture process, "embedded" into the group activities, has been initiated to verify the existing QA checks, define new ones and document all rationale. The resulting checks have been automated in a conventional software program to achieve the desired standardization, integrity and time reduction. A prototype electronic knowledge base has been developed with Macintosh's HyperCard™ to serve as a knowledge capture tool and a knowledge repository. It has also been designed to support a future capability to automatically generate code. The success of the effort has been demonstrated by the adaptation of the knowledge capture process by two other Ascent groups.

INTRODUCTION

This paper focuses on capturing quality assurance (QA) knowledge for the unique environment and needs that exist in the Flight Design and Dynamics Department (FD&DD) of the Rockwell Space Operations Company (RSOC). The quality assurance knowledge capture process began with a group within FD&DD that designs the post-MECO (Main-Engine Cutoff) phase of the space shuttle ascent trajectory. This post-MECO project is serving as the prototype and pathfinder for subsequent QA knowledge capture projects.

Faced with an increasing flight rate, FD&DD is looking for ways to increase productivity while containing the number of personnel. FD&DD consists of approximately 500 employees who design the ascent, orbit, and entry trajectories for shuttle missions. It produces approximately 500 products for each mission, performs related analysis and provides real-time support in the Mission Control Center. The current flight rate of about nine missions per year is expected to increase to twelve in 1992. QA within FD&DD currently consists of highly manual, repetitive, and time consuming tasks. Both the externally delivered products as well as tasks intermediate to generating products require QA. The development and use of knowledge tools is recognized to be one source of productivity improvements. Knowledge tools have potential not only to reduce task time, but also to reduce risk and increase quality in QA.

The following sections discuss in detail the significant aspects of the project to date as well as future plans. First, several problems uncovered early in the knowledge capture effort are discussed. Discussed second are the knowledge capture process that evolved and the results, the knowledge tools. Presented next is the knowledge-based management system (KBMS) along with its utility for on-going knowledge maintenance, and other applications of the KBMS knowledge. In conclusion, the self-sustaining nature of the knowledge capture methodology, e.g., the knowledge capture process and resulting knowledge-base tools, is presented along with plans to apply the methodology in other areas.

THE KNOWLEDGE CAPTURE PROBLEM

The post-MECO QA task was selected as the initial QA task to be automated in FD&DD because of the potential high manpower reduction, and the enthusiasm and support of key group members. Post-MECO QA centers around the Space Vehicle Dynamics Simulator (SVDS), a six degree-of-freedom simulation of shuttle flight dynamics. The Post-MECO group uses SVDS to perform validation of initialization variables (I-loads) loaded in the onboard software for each mission. SVDS executes nominal post-MECO and abort scenarios using the I-loads in the input runstream. Significant flight-related parameters are output at specific events during the simulation. Group members "QA" the simulation by first detecting errors revealed in the output, determining their cause, and then determining the correction, analogous to fault detection, isolation, and resolution (FDIR). Errors are detected by applying boundary-value constraints to the SVDS output. Knowledge of the problem and simulation software are then used to determine the cause and to provide a fix.

Initially, capture of the QA knowledge involved some significant concerns, the environment and condition of the knowledge. The QA information passed on to FD&DD at the beginning of the Space Transportation Systems Operations Contract (STSOC) consisted only of a skeleton checklist of boundary-valued constraints for error detection. Rationales for the checks and problem resolutions were not documented. Although some of the experienced personnel relocated at STSOC, most of the post-MECO domain specialists were in the process of acquiring expertise.

To train the domain specialists the skeleton checklist was used with a large verbal transfer of knowledge. When an inexperienced engineer needed to resolve a violation of the checklist, assistance was often necessary. This assistance was provided by an experienced group member, the expert, who had knowledge in that specific problem area. The engineer did not always learn the rationale behind a constraint until he/she had to resolve a violation. As the engineers obtained experience, they improved their checklists, and the individual checklists diverged. However, there could be checks on a list which the engineer did not fully understand. The engineer performed these checks by rote, and for these checks worked with a "black box".

Due to heavy workloads of the experts, they were not always available. If the experts were unavailable, another method of resolving a violation was to determine the rationale of the constraints from formal documents. The formal documents are texts on flight design, flight rules, vehicle constraints, groundrules and constraints, and flight software requirements. There is then the problem of determining which document to search, locating it, and finding the specific information within it. Therefore, capture of the rationales would require much research and time. The heavy workload of the engineers limited the time any one of them could spend on research.

THE KNOWLEDGE CAPTURE APPROACH

After identifying the condition and environment of the knowledge, the knowledge capture approach required would

- Capture the dispersed rationales and problems resolutions.
- Support the in-progress training of the domain specialist engineers.
- Fit within the discipline specialists time constraints.
- Support long-term, continuous capture.
- Easily incorporate changes in the QA knowledge as the group's novices become experts (Reference 1).

- Prevent any resulting automation aids from becoming a black box, i.e., support knowledge retention.

- Support and speed-up training of future new-hires, expected because of routine turnover and additional workload caused by the increasing flight rate.

- Reduce the workload of the available knowledge engineers to free them for pending projects.

It was decided that these objectives could best be met by making knowledge capture an integral part of the group's work process and, if possible, by making it self-sustaining.

Because of the experience level of the engineers and the uncertainty about the return on automating problem resolution, it was also decided to divide the knowledge capture into two phases. Phase I would capture the error detection criteria used in the QA of the simulation runs and the rationales behind them. Phase II would focus on the problem resolution knowledge, i.e., determining the cause of an error and deciding on a fix. The first part of the effort in Phase I would focus on obtaining a consolidated list of boundary-value constraints. Immediate benefit could be realized by the group by automating these checks. A straight-forward FORTRAN program would be written, and used during Phase I. The second part of Phase I would concentrate on capturing constraint rationale and refinement of error-detection constraints; the refined checks would be used to update the FORTRAN program.

One person would not be able to gather all the lost knowledge due to the dispersed nature of the knowledge and personal time constraints. To overcome this problem, a divide-and-conquer approach was taken. This approach required involvement of all members of the Post-MECO group. Existing group meetings were used to consolidate the error checks and then to recover the rationales. Once the errors checks were consolidated, members began to systematically discuss rationales in the same order the checks are made during the run. If a rationale was not obtainable from some member of the group, a specific member was assigned the task of researching the rationale and recording his/her results on a form created for this purpose. On the form, the discipline specialist filled in the criteria information, what the check did and under what circumstances it was valid, and detailed the rationale of the criteria. A reference was also cited, those of the formal texts when possible and historical data if formal reference documents did not define the constraint or rationale. The historical data referred to previous 6-DOF SVDS output that was available from recent missions. This provided referential validation of the knowledge, both the constraints and their rationale (Reference 2). At a subsequent meeting the rationale was scrutinized by the group, aiding the group learning process and refining the information. The completed form was kept in a notebook for further reference and use.

To support the group process and off-load time consuming acquisition of group expertise by the knowledge engineer, a project lead was recruited. The project lead's functions were to

- Evaluate the captured knowledge and determine deficiencies.

- Ensure the clarity of the recorded knowledge.

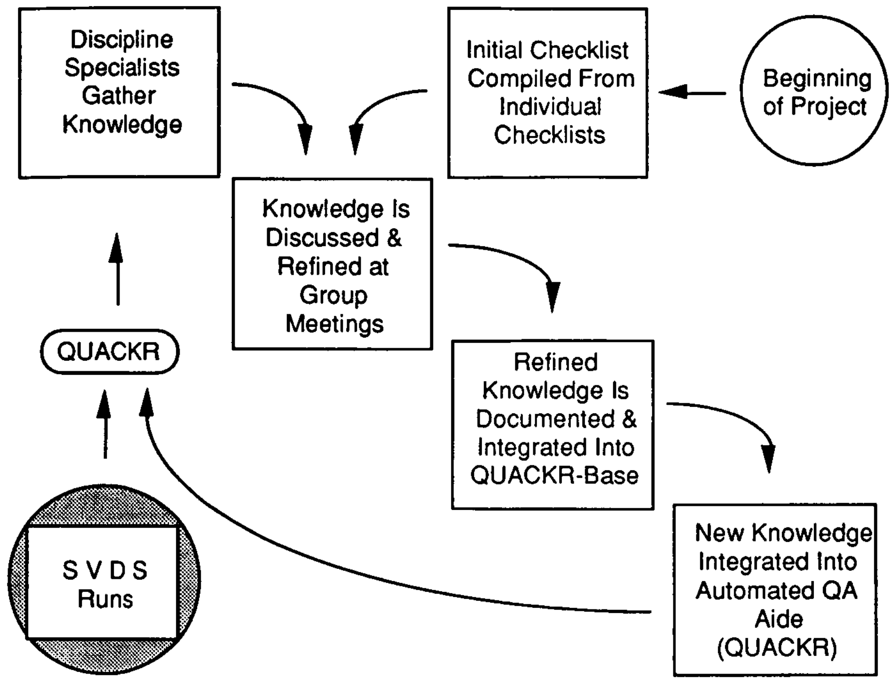
- Set the agenda for meetings.

- Hand out research assignments and follow-up to see that they are accomplished.

- Maintain a notebook that consolidates the captured knowledge.

Some initial training in group dynamics and knowledge capture was needed to prepare the project lead for these responsibilities. This training was provided by the knowledge engineer. After the process was underway, the knowledge engineer needed only to advise the project lead in these areas.

The approach used for the knowledge capture results in a self-perpetuating refinement process, Figure 1. As QUACKR (Quality Assurance Checker), the FORTRAN program is used, the error criteria and their rationale are placed under further scrutiny. In the course of their examinations, the discipline specialist gather more knowledge. Additions and improvements to the knowledge are brought into the group meetings by group members. The knowledge is refined during group discussions. When a consensus is reached and the information is comprehensible to the knowledge engineer the subjective validation of the knowledge has also occurred (Reference 2). This improved knowledge is added to the knowledge-base, QUACKR-base, and used to upgrade QUACKR.



Discipline Specialists Gather Knowledge
Initial Checklist Compiled From Individual Checklists
Beginning of Project
Knowledge Is Discussed & Refined at Group Meetings
QUACKR
Refined Knowledge Is Documented & Integrated Into QUACKR-Base
S V D S Runs
New Knowledge Integrated Into Automated QA Aide (QUACKR)

A Self-Sustaining Knowledge Capture Process

Figure 1

Interest in the knowledge capture process has been sparked by the potential of QUACKR. The member's of the Post-MECO group realize that the rationales are needed to validate QUACKR, and they were eager to carry out their assignments. Capture was completed in December, 1989, marking the end of Phase I. The rationales are accessible from the QUACKR-base to serve as off-line explanations to QUACKR. The documentation and accessibility of the QUACKR-base is an excellent improvement over the previous condition of the information. Through their experiences, the group has discovered that the rationales also give a good head start toward finding a resolution.

Along with improving the knowledge in QUACKR, the individuals in the group have improved their own knowledge at an accelerated pace. Discussing the knowledge at meetings not only refines the knowledge, but also passes the knowledge on to other members of the group. The group also has gained "meta-knowledge": they know more about what they know. Individuals had developed areas of specialization and these specialists became identifiable during the group meetings.

McGraw and Harbison-Briggs (Reference 3) emphasize the importance of maintaining good, working relationships between the knowledge engineer and the group and between members of the group. The project lead has established working relationships with the group which are extended to the knowledge capture context. The project lead is also familiar with the relationships among the experts. This has lessened the knowledge engineer's concern over these relationships.

KNOWLEDGE TOOL

As a post processor to the SVDS run, an expert system, QUACKR (Quality Assurance Checker), which checks boundary constraints was built. It is a FORTRAN program that uses the defined and verified list of boundary constraints to detect violations and print warning messages.

QUACKR is the most obvious physical results of the knowledge capture process. Most important about this knowledge-based aid (KBA), even though it is conventionally implemented, is that it provides consistent, rigorous performance of the fault detection task. QUACKR began with approximately 300 checks accumulated from the group's checklists. The group knowledge capture process has resulted in the addition of over 100 checks. The expert's schedules do not allow them to perform the 400 time-consuming checks manually. QUACKR performs the evaluations in three minutes of UNIVAC 1192 computer time. As the knowledge in QUACKR is upgraded, the time savings will increase. As the flight rate increases, the use of QUACKR will increase, thereby increasing the cost savings QUACKR provides.

THE KNOWLEDGE-BASE MANAGEMENT SYSTEM

The electronic KBMS, written in HyperCard, will eventually replace the project leader's notebook as the place to maintain and access the knowledge. The knowledge-base is referred to as the QUACKR-base. Eventually the QUACKR-base will include all error detection constraints and their rationale (Phase I), causes of constraint violations and resolutions (Phase II). The QUACKR-base provides a user-friendly means for the group to access the knowledge. This allows the group to

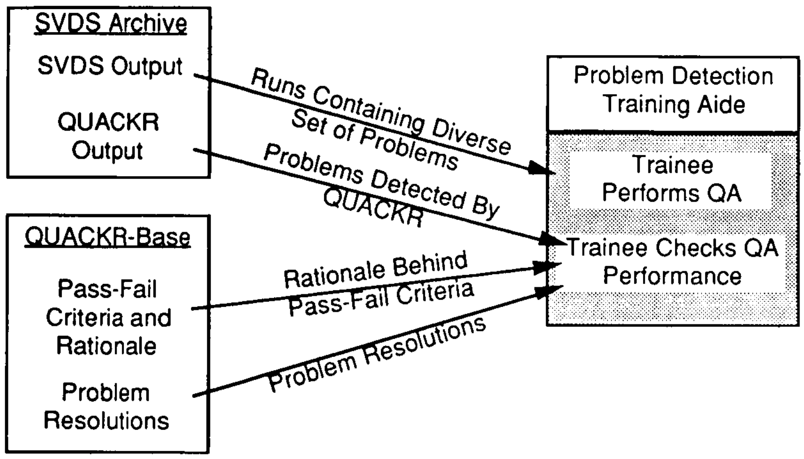
- Simultaneously maintain Quackr and Quackr-base
- Capture knowledge with currently undetermined structure
- Support changes to knowledge easily
- Remain cognizant of the knowledge utilized by QUACKR, preventing it from becoming a "black box".
- Support new hire training.
- Reference information pertinent to problem resolution.

The knowledge is stored in the knowledge-base in a format which is much more readable than FORTRAN code. Keeping the QUACKR-base separate from QUACKR would introduce a point in the process where the knowledge in QUACKR and that in the QUACKR-base can become mismatched. The code generation capability prevents this from occurring. This utility allows the user to generate new QUACKR code whenever refinements are made to the knowledge in the QUACKR-base, keeping the knowledge in QUACKR and the QUACKR-base consistent. The validity of the knowledge in the knowledge-base, therefore, reflects the validity of the knowledge in the KBA. Verification of the knowledge in the KBMS, both QUACKR and QUACKR-base, is made simpler through the use of consistency and completeness routines written in HyperCard's scripting language.

Another reason for choosing the HyperCard platform is that it allows a developer to create different structures for the various types of knowledge. This flexibility allows additional knowledge types to be included in the knowledge-base and allows changes in the current structure of knowledge. For example, parameter knowledge is stored as objects and checks are stored as if-then statements. Since the capture of problem resolution knowledge has not begun yet, the appropriate structure for storing that knowledge is unknown. However, as mentioned previously, the rationale behind a check often points to resolutions should that check fail. Consequently, some problem resolutions have been captured. Currently, these are stored in variable length text fields. After more are captured, an appropriate structure will be decided upon, and the problem resolutions will be stored in the QUACKR-base along with the problem detection knowledge. To exemplify how the knowledge may change, as the novices in the group become experts, their method of problem resolution may become more schema-driven than data-driven (Reference 1). The flexibility of the KBMS will also allow extensions to restructure the knowledge.

As shown, the KBMS is easily extended to contain additions to and changes in the knowledge. The new and refined knowledge can in turn be used to develop other automated aids, i.e., a rule-based problem resolution expert system, which will be managed using the KBMS. The hyperlink capability of HyperCard allows links to be built between the types of knowledge so that logical connections between the types are maintained. In this way the hyperlinks also define the relationships between the knowledge tools.

Since the QUACKR-base contains the relevant knowledge it stands alone as an excellent reference source. The QUACKR-base is not readily available to the users currently, so, it is more convenient to parse the knowledge-base and create a reference document. The KBMS has report utilities which will do this. The knowledge in the QUACKR-base will also benefit training. Presently, trainees learn experientially. Various scenarios are being considered which utilize the knowledge in the QUACKR-base for training, such as the one in Figure 2.



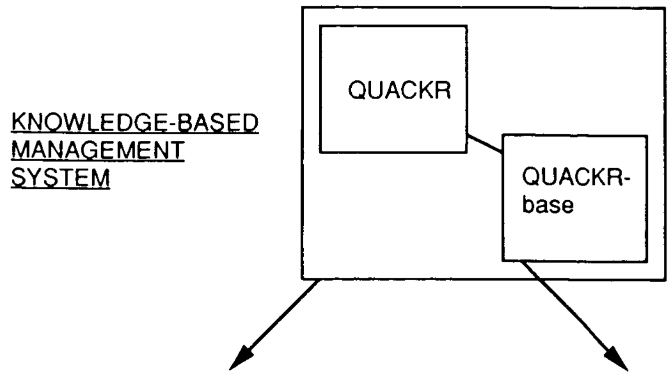
SVDS Archive
SVDS Output
QUACKR Output
QUACKR-Base
Pass-Fail Criteria and Rationale
Problem Resolutions
Runs Containing Diverse Set of Problems
Problems Detected By QUACKR
Rationale Behind Pass-Fail Criteria
Problem Resolutions
Problem Detection Training Aide
Trainee Performs QA
Trainee Checks QA Performance

A Scenario For Training Using The QUACKR-Base

Figure 2

This training scenario involves compiling a set of SVDS runs with a diverse set of problems and having the trainee detect faults manually. The trainee could check his/her own performance against the QUACKR output. The trainee could also resolve any problems manually and match his/her performance against resolutions in the QUACKR-base. This scenario would provide more in depth training, with less demand on experienced personnel, in a shorter amount of time than the present method of training. By training with the knowledge in the knowledge-base, the group remains aware of what knowledge is in QUACKR and the rationale behind the QA process, which keeps QUACKR from becoming a "black box."

The KBMS satisfies many objectives of the project with its capabilities which manage the acquired QA knowledge and the knowledge tools. Figure 3 depicts the functions and capabilities of the KBMS.



Management functions	Capabilities
• Autonomous management of QUACKR	Automatic code generation
• Support long-term continuous capture	Flexible knowledge structuring
• Easily change QA knowledge when necessary	User friendly platform
• Support knowledge retention	Contains the QA knowledge and provides an easy reference source
• Provide QA knowledge documentation	Report generation of QA knowledge
• Support training	Reference source and documentation

KBMS Functions and Capabilities

Figure 3

CONCLUSIONS

It is possible for the Post-MECO group to sustain their own knowledge capture process and maintain the knowledge-based management system themselves. Two things lead us to believe this is possible. One is the project lead. Once the knowledge capture process is started, the knowledge engineer becomes a consultant to the project lead. The Post-MECO group is already capturing their knowledge self-sufficiently. Maintaining the QUACKR-base should be easy enough that anyone can do the maintenance. However, only one person from the group will be trained as a QUACKR-base manager and allowed to update the QUACKR-base knowledge for configuration control reasons. The knowledge engineer will remain in charge of upgrading the QUACKR-base software when needed. With the consistent effort of the QUACKR-base manager to update the QUACKR-base with current knowledge, the automatic effortless export of knowledge from the QUACKR-base into QUACKR will satisfy the maintenance requirement of the knowledge-based system.

This self-sustaining knowledge capture process and resulting KBMS fulfills the objectives of this task. Through the approach chosen the dispersed rationales were captured while supporting the in-progress training of the engineers. The rationale refinement meetings were within the discipline specialists time constraints and were able to increase the level of knowledge within the group. With the help of the project lead the knowledge engineers were given some relief from the project.

The reasons for doing this task were accomplished with the knowledge tools. The knowledge tools reduced the QA task time sufficiently and also documented the available and acquired QA knowledge. The availability of this knowledge to an engineer should prevent QUACKR from becoming a black box.

The management of this knowledge is provided by the capabilities of the KBMS, QUACKR and QUACKR-base. Long-term continuous capture can be maintained from the continuous generation of QUACKR output and the flexibility of the knowledge structure within QUACKR-base. Changes to the knowledge are easily accommodated because an easy to use system was chosen, Hypercard. Knowledge retention and training are supported with report generation and the accessibility to a user friendly reference source, the QUACKR-base.

Because the prototype project has been successful in attaining the time reduction of the QA task and has also been able to meet many other objectives this process has begun in two other areas within FD&DD. These new groups have begun with the compiling of a master checklist and capturing rationale. This second effort will be based on what was learned during the Post-MECO QUACKR project and the unique needs of this group. There are at least ten areas within FD&DD that can benefit from the type of knowledge capture process and resulting knowledge-based systems that have evolved from the Post-MECO experience. Knowledge-based systems are planned to be in place by 1992 to support the increasing flight rate.

REFERENCES

1. LaFrance, Marianne, "The Quality of Expertise: Implications of Expert-Novice Differences for Knowledge Acquisition". A CM Newsletter, Special Issue On: Knowledge Acquisition, No. 108, April 1989.

2.Boose, John and Shaw, Mildred. "IJCAI-89 Tutorial Notes: Knowledge Acquisition for Knowledge-Based Systems," Detroit, Michigan, 1989.

3.McGraw, Karen and Harbison-Briggs, Karan. Knowledge Acquisition Principles and Guidelines. Prentice Hall, New Jersey, 1989.

N91-20692

QUANTITATIVE KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS

Brenda L. Belkin* and Robert F. Stengel**

Princeton University
Department of Mechanical & Aerospace Engineering
Princeton, NJ, 08544

ABSTRACT

A common problem in the design of expert systems is the definition of rules from data obtained in system operation or simulation. While it is relatively easy to collect data and to log the comments of human operators engaged in experiments, generalizing such information to a set of rules has not previously been a straightforward task. This paper presents a statistical method for generating rule bases from numerical data, motivated by an example based on aircraft navigation with multiple sensors. The specific objective is to design an expert system that selects a satisfactory suite of measurements from a dissimilar, redundant set, given an arbitrary navigation geometry and possible sensor failures. This paper describes the systematic development of a Navigation Sensor Management (NSM) Expert System from Kalman Filter covariance data. The development method invokes two statistical techniques: *Analysis of Variance (ANOVA)* and the *ID3 algorithm*. The ANOVA technique indicates whether variations of problem parameters give *statistically* different covariance results, and the ID3 algorithm identifies the *relationships* between the problem parameters using probabilistic knowledge extracted from a simulation example set. ANOVA results show that statistically different position accuracies are obtained when different navigation aids are used, the number of navigation aids is changed, the trajectory is varied, or the performance history is altered. By indicating that these four factors significantly affect the decision metric, an appropriate parameter framework was designed, and a simulation example base was created. The example base contained over 900 training examples from nearly 300 simulations. The ID3 algorithm then was applied to the example base, yielding classification "rules" in the form of *decision trees*. The NSM expert system consists of seventeen decision trees that predict the performance of a specified integrated navigation sensor configuration. The performance of these decision trees was assessed on two arbitrary trajectories, and the performance results are presented using a predictive metric. The test trajectories used to evaluate the system's performance show that the NSM Expert adapts to new situations and provides reasonable estimates of sensor configuration performance.

INTRODUCTION

Knowledge acquisition is a major problem in the development of rule-based systems. The tools developed to date are not designed to extract information from data for which no generalizations are known *a priori*. Instead, these tools either rely on the expert to provide examples from which rules are generated or try to capture the expert's problem-solving methodology with interviewing techniques [1]. Unfortunately, it often is difficult for experts to describe their problem-solving methods or to detail the factors that come into play during the resolution of a problem. It is exactly this type of knowledge that is needed to design rule-based systems.

Since the early 1970's adaptive navigation has been viewed as a highly desirable candidate for development in next-generation aircraft [2]. It is envisioned that future aircraft will have multi-sensor capability for navigation tasks requiring high reliability, optimal performance, and increased automation. With multi-sensor capability, the task of sensor configuration selection and management will become an additional pilot burden.

The performance of multi-sensor navigation systems (more commonly known as "integrated" or "hybrid" systems) has been explored since the late 1960's when results from modern control theory provided techniques for sensor mixing and optimal state estimation [3]. Hybrid systems refer to externally referenced navigation systems that "aid" an on-board inertial navigation system (INS) using an optimal state estimation mechanization. Hybrid navigation systems combine the high- and low-frequency accuracy properties of INSs and external navigation aids (navaids) respectively. Many radio navigation and on-board systems aiding INS have been modelled and their performance covariance results obtained [4-8]. When radio navigation systems are only partially operational, results show that improved navigation performance is obtained over that of the pure INS [4]. Therefore it becomes advantageous to keep partially operational systems as candidates for integrated sensor mixing purposes.

With a large number of available navaids, choosing an optimal or near-optimal sensor set becomes a large combinatorial problem. Convergence towards an optimal sensor configuration requires an exhaustive computer search utilizing simulation results as the basis for selection. In contrast, a small number of available navaids reduces the decision space considerably. Hence, a dilemma occurs; increasing sensor capability (and thus reliability and performance) increases decision-making complexity.

The selection of an optimal configuration requires the application of some decision criteria. Most often, designers choose between navaids based on the relative accuracies of each system using a hierarchical approach [9]. This approach is "knowledge-based" in the sense that the nominal performance of the systems is well-known and that this knowledge is built into the sensor hierarchy. The current hierarchical designs are not as "robust" with respect to sensor availability and performance changes as is necessary for future sensor management systems [10]. Instead, these hierarchies represent "rules-of-thumb" that

* Formerly, Graduate Student, Princeton University, Currently, Member of Technical Staff, AT&T Bell Laboratories, 480 Red Hill Road, Middletown, NJ, 07748

** Professor of Mechanical & Aerospace Engineering are useful in only the simplest cases. They do not resolve sensor configuration problems when more detailed information must be considered – for example when the number of each available navaid is specified, when partially operational systems remain viable candidates, and when trajectory effects degrade system performance. It becomes necessary to explore factors other than the performance of nominally operating navaids to determine how these factors affect the decision-making process, and to exploit the potential of hybrid systems.

The statistical technique Analysis of Variance (ANOVA) [11] was used to identify the factors that cause variation in navigation performance. Once the important factors were identified, the relationships between them were determined. The ID3 algorithm [12,13], an inductive inference technque based on the probabilistic occurrence of events, was used to find these attribute relationships.

The development of a navigation sensor management expert system using the ANOVA/ID3 technique [14] is described in this paper. The NSM system controls the selection of multi-sensor configurations. The methodology is applicable to any problem where the development of knowledge bases from multi-factor data studies is desired.

INTEGRATED NAVIGATION SYSTEMS

Optimal estimation techniques are used to combine inertial and radio navigational systems in order to provide stable continuous inertial navigation information [15]. The errors exhibited by these "hybrid" systems depend on the accuracy of the aiding system, and navaid accuracies are functions of many factors such as navaid type, number of similar navaids, and trajectory parameters such as distance from the navaid and whether the aircraft is approaching or receding from the station. The sensor selection criteria depend on the relative importance of these factors. Five external radio navigation and two on-board navaids were used to update a medium-accuracy (10 N. Mi/hr) INS. Hybrid system performance was simulated using the linearized inertial navigation error model and navaid measurement models as inputs into the optimal estimation filter. The hybrid errors were updated at a specified navaid fix rate. The systems simulated were (1) Global Positioning System (GPS), (2) Long-Range Navigation System (LORAN), (3) Tactical Navigation System (TACAN), (4) Distance Measuring Equipment (DME), (5) VHF Omnidirectional Range (VOR), (6) Doppler radar, and (7) air data sensor. The operational theory and the mathematical models used to simulate the navaids and the inertial navigation error model are discussed in detail in [14].

The numerically-stable discrete-time U-D implementation of the Kalman Filter equations was used to mix the inertial system and navaid information optimally, providing covariance estimates of the navigation errors (e.g., north/east position) [14,17]. Each nonlinear measurement equation was linearized with respect to the inertial navigation states to obtain the observation matrix used in the U-D measurement update equations. Since sensor errors were taken into consideration in the measurement models, the inertial error state vector was augmented with the sensor shaping filter dynamics (e.g., random bias, first-order Markov model) to formulate the hybrid navigation model. Additionally, the measurement noise time history was simulated. As the aircraft moves along its trajectory relative to ground-based navaid stations, the measurement noise characteristics change. Therefore an equation for a distance- or time-varying measurement covariance matrix was found in order to realistically model ground-based radio navigation systems. According to Ref. 17, GPS measurement noise increases in a similar way; as the satellite descends near the aircraft's horizon, the noise increases. To simulate time-varying measurement noise for the ground- and satellite-based navigation systems, each noise variance was modelled as the sum of initial and range-dependent variances. The latter component increases linearly with the square of the distance from the station or satellite.

Position accuracy was selected for the rule-based system decision metric. Here, position accuracy is defined as the root sum of squares (RSS) of the north and east component errors. The RSS decision metric provides sufficiently consistent quantities to compare hybrid performances. For a detailed discussion of the RSS decision metric, the reader is directed to Ref. 14.

HYBRID NAVIGATION SIMULATION RESULTS

Using the RSS position error metric to measure hybrid system performance, the following U-D filter simulations were performed:

1. Single-type hybrids: GPS, LORAN, TACAN, DME, VOR, Doppler Radar, or Air Data Sensor aiding an INS
2. Number of stations used in a single-type hybrid
3. Multi-type hybrids: Combinations of different navaid types aiding an INS
4. Aircraft trajectories simulated: High-performance, commercial, general aviation

Comparisons of Single-Type Hybrid Performance

Consider the four ground stations A, B, C, and D spatially oriented with respect to the high-performance, commercial, and general aviation trajectories in Fig. 1. The four ground stations are simulated as LORAN slaves, TACAN, DME, or VOR stations. Figure 2 shows the performance differences of ground-based, GPS, and on-board type hybrid systems. When the results from all ground station A types (LORAN, TACAN, DME, VOR) are compared on the high-performance trajectory, the relative performance from best to worst may be listed as follows: (1) LORAN, (2) TACAN, (3) DME, and (4) VOR. For example, a hybrid system utilizing LORAN slave station A provides better performance than a hybrid system utilizing TACAN A; a TACAN A hybrid in turn outperforms a DME A hybrid which in turn outperforms a VOR A hybrid. This pattern is repeated for stations B, C, and D [14]. The best hybrid performance was obtained from three GPS satellites aiding the INS. Figure 2 also shows how the performances of the Doppler radar hybrid and the air data sensor hybrid compare with the GPS and ground-based navaid hybrids.

Referring to the LORAN results in Fig. 3, there is a striking variation in the performance of the individual Stations A-D; this figure reveals that single stations of the same type aiding an INS give highly variable performance results. The same variability in performance of the remaining ground-based single-station navaids was found [14]. From Fig. 3, the variation in Station A-D's performances is attributed to the position of each ground station relative to the aircraft's trajectory. For example, LORAN Slave A gives the smallest position error of the four stations; referring to Fig. 1, the aircraft makes a close approach to Slave A on the trajectory's second leg. Hence the RSS error becomes very small. These errors begin to increase towards the end of the trajectory leg, due to the increasingly uncertain north component. In contrast, LORAN Slaves B, C, and D are farther from the aircraft's trajectory. The first trajectory leg results in good relative north information to B, C, and D, whereas the east component uncertainty grows due to the lack of relative east information. The variations in performance observed from Stations A-D are due to trajectory effects; using Station B instead of A to update

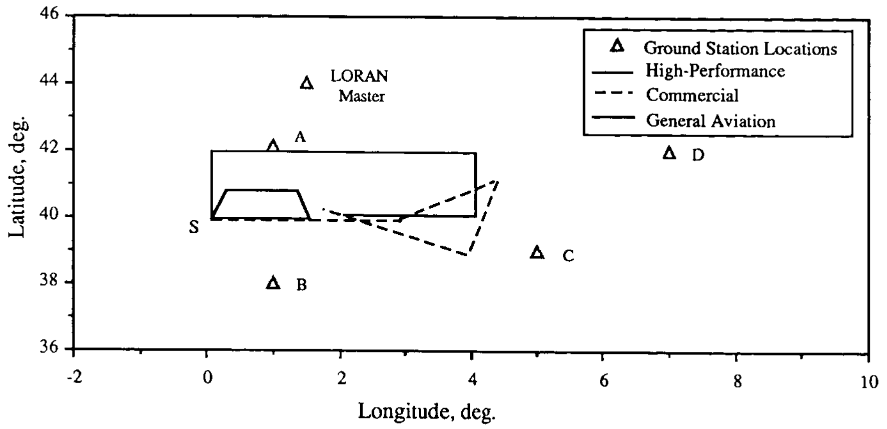


Figure 1. Aircraft Trajectories Used in Simulations

the INS is equivalent to using A and changing the aircraft's trajectory.

Effect of Increasing the Number of Navaids in a Hybrid System

Next, the effect of the number of ground stations was studied by simulating all possible combinations of single, double, and triple stations formed from stations A-D. There are six possible combinations of two stations and four combinations of three stations that may be integrated to aid the INS. These simulations were carried out for LORAN, TACAN, DME and VOR.

Referring to the LORAN results in Fig. 4, the performance variation among the double station combinations and triple station combinations is less pronounced than the single station variations. The magnitude of the RSS errors decreases dramatically when two stations are used instead of one station. The RSS errors decrease further when three stations are used, although the magnitude differences are not as great. The reason why the RSS magnitudes of the double- and triple-station combinations are much lower is that the aircraft receives the best navigation information available. This also explains why there is much more variation in the results for the double station combinations than for the triple stations. Similar performance trends were observed for GPS, TACAN, DME, and VOR [14].

Effect of Trajectory on Hybrid Performance

It already has been shown that an aircraft's trajectory relative to a single ground station hybrid plays an important role in the estimator's performance. The RSS results in Fig. 5 illustrate the performance differences of the LORAN Slave A hybrid on the high-performance, commercial transport, and general aviation trajectories. Two parameters that contribute to these performance differences are distance to a station and heading with respect to a station. A third trajectory parameter that contributes to a hybrid system's performance is the number of heading changes along the trajectory. The effect of heading changes is discussed in more detail in [14]. Trajectory factors affect the INS dynamics, which in turn affects the error estimation performance. The trajectory factors also change the measurement dynamics since the measurements are dependent on the trajectory's geometric properties and aircraft states (such as velocity). The results in Fig. 5 clearly show that when the trajectory changes, the navaid selection decision most likely changes as well since the relative accuracies of the navaids change.

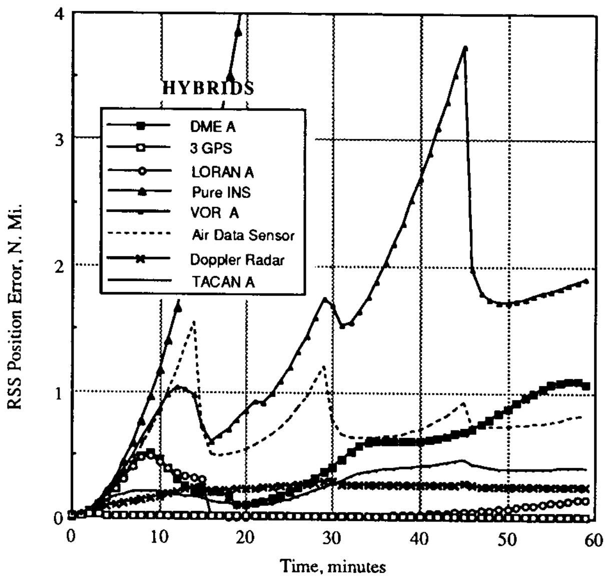


Figure 2. Performance of Satellite, Ground Station-Based and On-Board Hybrid Navigation Systems

Hybrid Performance of Mixed Navaids

Figure 6 shows various combinations of integrated navaids. The individual performances of LORAN Slave B, Doppler radar, and Air Data hybrids are shown in Fig. 6 along the high-performance trajectory. The LORAN/Doppler and LORAN/Air data hybrids also are plotted in this figure for comparison. Both combinations gave better results than their individual components operating alone. For example, the LORAN/Doppler combination outperformed the LORAN hybrid and the Doppler hybrid; similarly, the LORAN/Air Data combination gave better results than did the LORAN alone or the Air Data sensors alone. The latter combination did slightly better than Doppler hybrid on this trajectory after the initial transient period. These results show that good navigation performance is still obtainable when a "failed" LORAN system (only one slave station operational) is integrated with an on-board navaid such as Doppler radar or a standard equipment air data sensor.

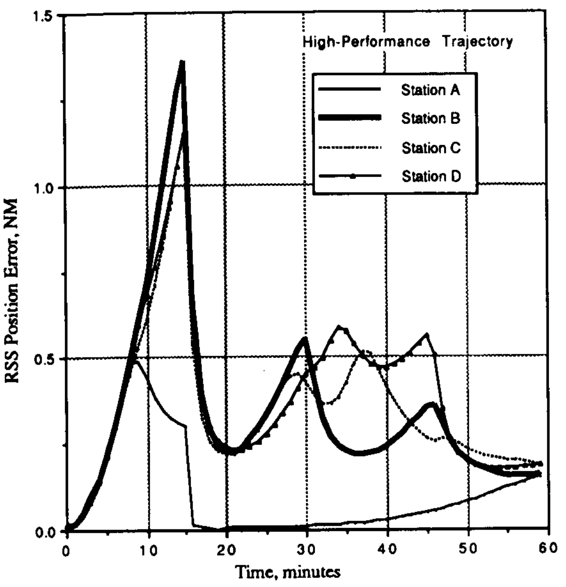


Figure 3. Performance of Single-Station LORAN Navaids Aiding an INS.

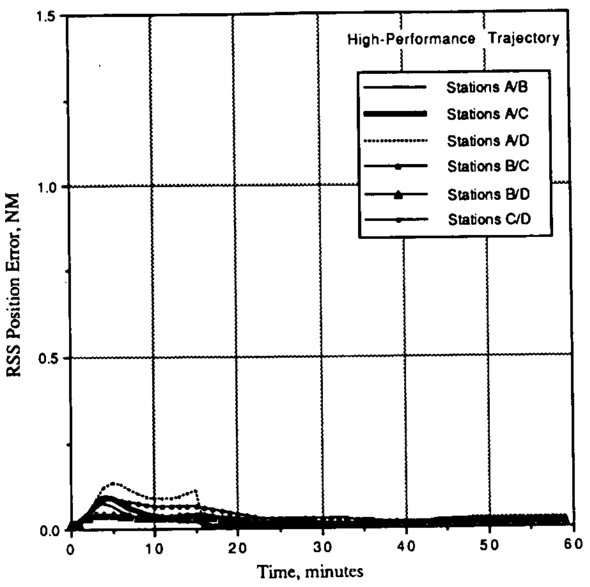


Figure 4. Performance of Double-Station LORAN Navaids Aiding an INS

DEVELOPMENT OF A NAVIGATION SENSOR MANAGEMENT EXPERT SYSTEM

This section describes a novel methodology that uses established statistical techniques to develop the NSM expert from the simulation data. The primary function of this expert system is to select the external navaid sensors that provide the smallest possible RSS position error from a large set of available sensors. The Analysis of Variance (ANOVA) technique [11] is used to identify the factors that make statistically significant contributions to the decision metric. Then, the ID3 algorithm determines the relationships between these factors [11,13].

Identifying Important Factors Using ANOVA

The ANOVA technique was applied as follows: first, the mean value of the RSS position error and the variance for all the simulations were computed. The ANOVA model decomposes the variance into a sum of variances, each associated with a potentially contributing factor. Over two hundred simulations were performed, and the data were used in a four-factor navaid experiment. The goal of the experiment was to identify which of the factors (navaid type, number of ground stations, trajectory effects, performance history) and their interactions had statistically significant impacts on the RSS position error. The factor states used in the ANOVA experiment were: Navaids={VOR, DME, LORAN, TACAN, GPS}; Number of Ground Stations={One, Two, Three}; Trajectory Type={High-Performance, Commercial Transport, General Aviation, from Fig. 1}; Time Interval = {I, II, III, IV}. Since each trajectory consists of four, fifteen-minute legs, the "Time Interval" factor refers to the RSS performance obtained within each fifteen minute time frame. Four single-station, six double-station, and four triple-station hybrids were simulated using combinations of Stations A-D in Fig. 1.

The ANOVA results [14] show that three of the four factors are strongly significant with 99% confidence; the fourth factor, trajectory, was shown to be weakly significant (90% confidence). The latter result suggested that additional investigation into the effect of trajectory on RSS position error is necessary for more specific trends to be observed. Indeed the term "trajectory" is extremely vague; the results from Scheffé comparison tests suggest that "trajectory" should be decomposed into attributes that describe, in better detail, what these effects really are. For example, some trajectory attributes include distance from a station, airspeed, and whether the aircraft is approaching or receding from the station. Scheffé multiple comparison tests were applied to the navaid and number of ground station factors to identify the specific differences within each groups; for example, the RSS performance difference between GPS and TACAN, all other factors being equal, was statistically significant. On the other hand, the RSS performance difference between LORAN and TACAN with all other factors being equal, was not statistically significant. This means that a LORAN hybrid could perform better or worse than a TACAN hybrid, depending on the values of the other factors (e.g., number of ground stations). The multiple comparison test results yielded the same performance ranking depicted in the graphical results (e.g., Fig. 2), while utilizing the information content of a large number of independent simulations. Further investigation into the ANOVA interaction effects revealed that the ranking should be cautiously applied to single-station hybrids, since these are highly-sensitive to trajectory effects. The complete factor analysis results are given in Ref. 14. In summary, the ANOVA and Scheffé methods systematically identified trends in the simulation data without recourse to tedious graphical analysis.

Extracting Rules Using Induction: The ID3 Algorithm

The ID3 Algorithm uses inductive inference to extract rules [13] from a training set of examples. The problem space is described in terms of attributes, where each attribute is characterized by a set of values that define the possible "states." For example, in the previous section, the navaid type and number of ground stations were shown to be attributes affecting RSS position error. The attribute values for the factor "navaid

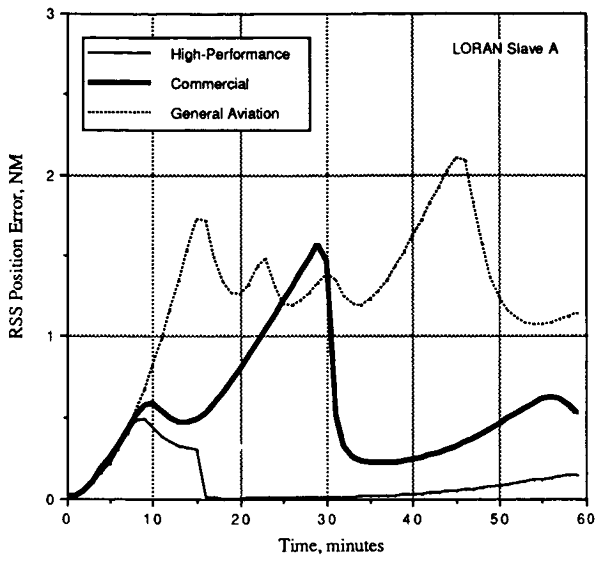


Figure 5. Comparison of RSS Results for LORAN Hybrids on Three Different Trajectories

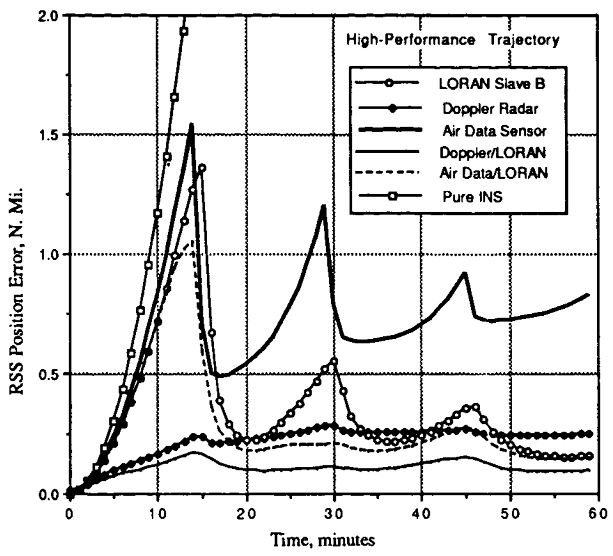


Figure 6. Comparison of RSS Results with LORAN, Doppler Radar, and Air Data Sensor Hybrid Combinations

type" were {GPS, LORAN, TACAN, DME, VOR}, and the attribute values for the factor "number of stations" were {One, Two, Three}. Hence there is a clear connection between ANOVA and ID3 problem structures. ANOVA factors are ID3 attributes, and ANOVA factor levels are ID3 attribute values.

An important problem in designing an inductive inference algorithm is identifying the attributes that span the problem space most efficiently, so that the resulting decision tree is as compact as possible. The ID3 algorithm selects the most important attributes using an information-theoretic measure (ITM) that minimizes the number of tests (attribute nodes) necessary to classify a problem. The ID3 algorithm utilizes a splitting strategy [12] to decide which attribute provides the most information from the example set. A detailed example illustrating how the splitting strategy is used to construct classification rules is given in [14].

Developing the ID3 Attribute Framework Using ANOVA Results

Up to three ground stations (four GPS satellites) were included as possible configurations. Time-weighted measurement effects are included in the attribute framework using RSS position error classification codes representing the hybrid's performance on a preceding trajectory leg. The trajectory effects were separated into the following attributes: geodetic distance from a ground station, line-of-sight angle from the station, and the direction of flight (approaching or receding) relative to a ground station. The distance from a ground station is an important attribute since the signal-to-noise ratio decreases as the distance to the station increases. The direction of flight with respect to the station influences position accuracy through its effect on the line-of-sight angle. The trajectory parameters were computed for each of the high-performance, jet transport, and general aviation trajectories on each trajectory leg. The maximum and minimum distances to the aiding station were also determined on each trajectory leg, in addition to the difference between the maximum and minimum distances.

When more than one station was used, the attributes were redefined slightly. The maximum and minimum distances then referred to the closest and farthest distances computed to the stations. The distance difference is the algebraic difference between the farthest and closest distances determined on the trajectory leg. A similar definition was applied to the line-of-sight angle; from the angles computed to each station, the largest and smallest were selected. The ID3 algorithm's task was then to determine how these attributes were related to each other and to the RSS performance.

The classification scheme chosen to represent the RSS position error endnode in the decision trees is depicted in Table I. Since an approximate prediction of the RSS position error was of interest, it was appropriate to represent the RSS performance in terms of an error range.

Table I RSS Position Error Classification Scheme

Accuracy					
[High]		[Medium]		[Low]	
Error (N. Mi.)	Code	Error (N. Mi.)	Code	Error (N. Mi.)	Code
0.00-0.02	c-1	0.10-0.20	c-6	1.0-1.5	c-15
0.02-0.04	c-2	0.20-0.30	c-7	1.5-2.0	c-16
0.04-0.06	c-3	0.30-0.40	c-8	2.0-2.5	c-17
0.06-0.08	c-4	0.40-0.50	c-9	2.5-3.0	c-18
0.08-0.10	c-5	0.50-0.60	c-10	3.0-3.5	c-19
		0.60-0.70	c-11	3.5-4.0	c-20
		0.70-0.80	c-12	4.0-4.5	c-21
		0.80-0.90	c-13	4.5-5.0	c-22
		0.90-1.00	c-14	> 5.00	c-23

The velocity, distance, and line-of-sight angles were expressed in terms of ranges instead of individual values, so that the expert system weights trends more heavily than specific examples. This renders the expert system more adaptable to new conditions, because matches between the actual and knowledge-base cases could be obtained more frequently.

The example set was developed using the attribute framework described above. The RSS position errors for each simulation were classified on each trajectory leg using the scheme in Table I. The ID3 example base was then created from each single-, double-, and triple-station simulation.

NSM Decision Trees

The NSM example set was divided into seventeen smaller example sets. The GPS and on-board navaid examples were grouped into one expert, whereas the ground-based navaid examples were divided according to navaid type and time (15-minute intervals). The ID3 algorithm constructed decision trees for each of the seventeen small expert systems that comprise the larger NSM Expert. The breakdown of the NSM Expert into smaller systems provides greater manageability of the training example base. The total number of examples used to develop the NSM Expert System was nine hundred and thirty-two. In total, two hundred and sixty Kalman Filter covariance simulations were performed to formulate the complete NSM example set. An additional thirty-seven simulations were performed to obtain a decision tree to estimate RSS performance when different navaid types are combined. The NSM expert system prompts the user for a set of flight conditions commensurate with the attribute/value lists used in the example set, and the resulting RSS classification code is returned to the user from the decision tree.

A typical decision tree obtained for the ground-based navaids is exemplified by the TACAN results. Figure 7 presents the decision trees for single-, double-, and triple-station combinations on the first fifteen-minute trajectory leg. Here, the majority of the testing nodes are trajectory parameters (distance, LOS angle, direction of flight with respect to the station(s)). The top or root node in Fig. 7 is the aircraft's direction of flight. This is expected because the distance and LOS angle attributes are dependent on directional motion. Distance, LOS angle, and groundspeed are results of the aircraft's motion, and hence, represent more specific problem parameters; therefore it is expected that these parameters appear at a lower depth in the decision tree. Figure 7 also shows that distance, ground velocity, LOS angle, and hybrid performance history are significant factors that enable a prediction of the RSS error to be made. The RSS classification results verify that the closer the aircraft is to a station(s), the smaller is the RSS error; other results show that the larger is the LOS angle, the smaller is the RSS error [14].

The expected performance of the GPS system on each trajectory leg is shown in Fig. 8. Note that the aircraft's groundspeed plays an important role in the GPS hybrid's performance. Velocity affects the measurement dynamics (history) and is therefore classified as a trajectory effect. From Fig. 8, the two-satellite hybrids are more sensitive to these velocity effects than are the three- and four-satellite hybrids.

Finally, the decision tree showing what position error range is expected when different navaid types are integrated in a hybrid system is presented in Fig. 9. Note that the decision tree is not specified for a given trajectory leg. The RSS position errors for these simulations were averaged over the entire flight time for the high-performance trajectory. The tree is organized in terms of the navigation method used: (1) Distance-Velocity (ρ-V), (2) Bearing-Velocity (θ-V), (3) Distance-Bearing (ρ–θ), (4) Distance-Distance (ρ–ρ), (5) Bearing-Bearing (θ–θ), and (6) Velocity-Velocity (V-V). These results show that LORAN is a better distance-measuring navaid than DME and that Doppler Radar is a better velocity-measuring system than the Air Data Sensor when ρ-V navigation is used. The ρ–θ results show that it is possible to obtain performance when LORAN and VOR are used. The LORAN/DME hybrid gives better results than two DME stations but worse performance than two LORAN stations. By far the worst results are obtained using two VOR stations. As discussed before, the VOR system is the least accurate measurement device of the seven systems studied, which greatly affects INS-VOR hybrid results.

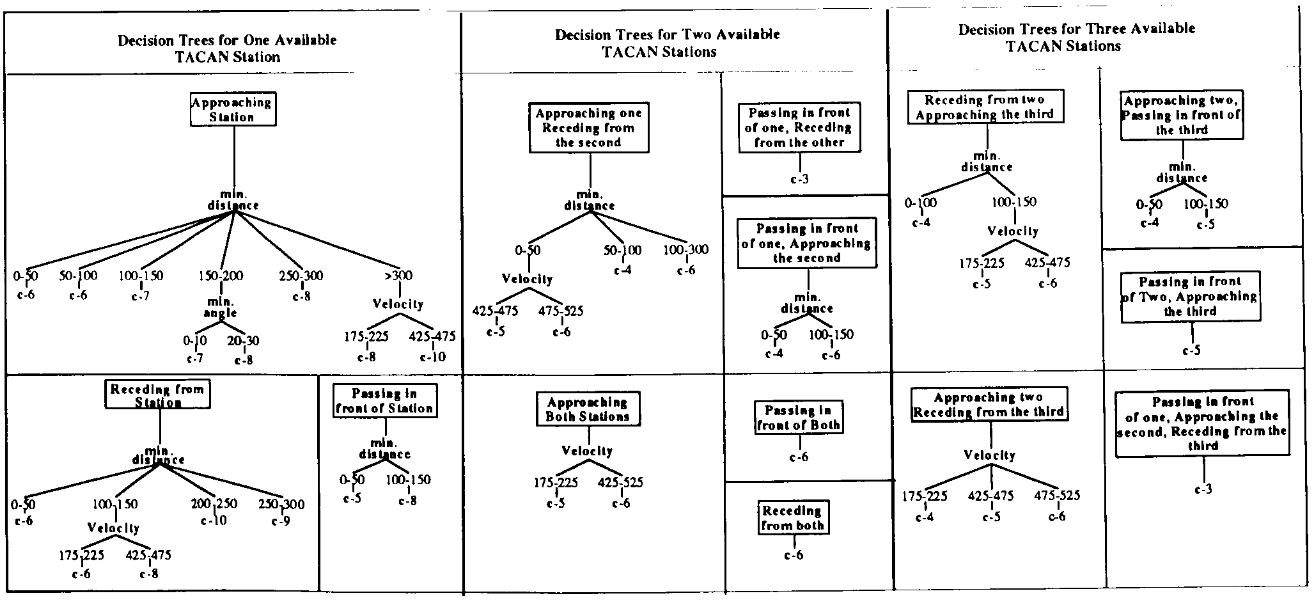

Decision Trees for One Available TACAN Station
Approaching Station
min. distance
0-50
c-6
50-100
c-6
100-150
c-7
150-200
min. angle
0-10
c-7
20-30
c-8
250-300
c-8
>300
Velocity
175-225
c-8
425-475
c-10
Receding from Station
min. distance
0-50
c-6
100-150
Velocity
175-225
c-6
425-475
c-8
200-250
c-10
250-300
c-9
Passing in front of Station
min. distance
0-50
c-5
100-150
c-8
Decision Trees for Two Available TACAN Stations
Approaching one Receding from the second
min. distance
0-50
Velocity
425-475
c-5
475-525
c-6
50-100
c-4
100-300
c-6
Passing in front of one, Receding from the other
c-3
Passing in front of one, Approaching the second
min. distance
0-50
c-4
100-150
c-6
Approaching Both Stations
Velocity
175-225
c-5
425-525
c-6
Passing in front of Both
c-6
Receding from both
c-6
Decision Trees for Three Available TACAN Stations
Receding from two Approaching the third
min. distance
0-100
c-4
100-150
Velocity
175-225
c-5
425-475
c-6
Approaching two, Passing in front of the third
min. distance
0-50
c-4
100-150
c-5
Passing in front of Two, Approaching the third
c-5
Approaching two Receding from the third
Velocity
175-225
c-4
425-475
c-5
475-525
c-6
Passing in front of one, Approaching the second, Receding from the third
c-3

Figure 7. Decision Trees Predicting RSS Position Error Range for an INS Aided by TACAN During the First 15 Minutes of Flight

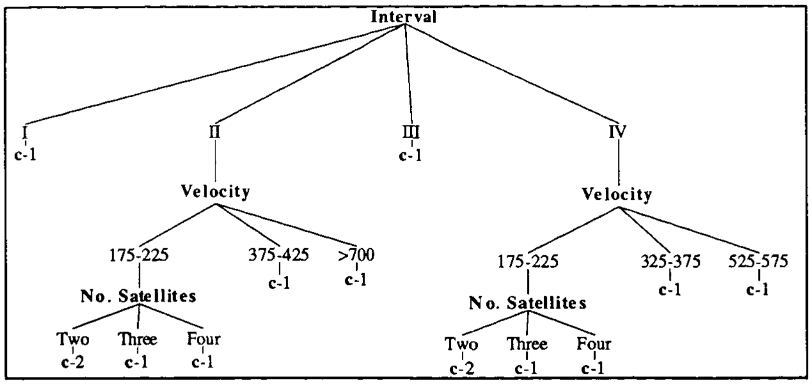


Figure 8 Decision Tree Predicting RSS Performance for an INS Aided by GPS

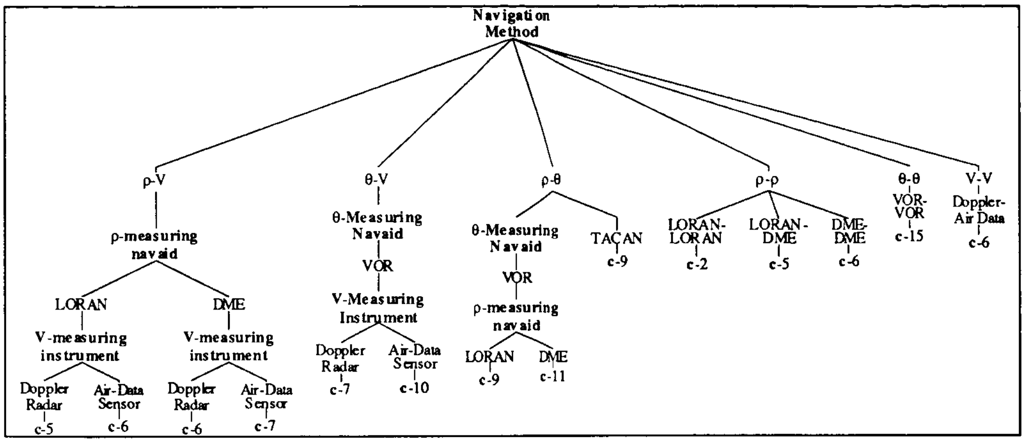


Figure 9 Decision Tree Predicting RSS Performance When Different Navaid Combinations are Used to Aid an INS

PERFORMANCE RESULTS OF NSM EXPERT SYSTEM

It is important to quantify the NSM Expert's performance for several test scenarios, in terms of how well it predicts a given hybrid's RSS position error. It is also important to gain insight into the factors that affect the system's performance, so that these factors can be exploited in future system development.

Two high-performance trajectories were used in the performance evaluation of the NSM Expert. The two trajectories each consist of four fifteen-minute legs. Trajectory #2's flight pattern was in a counter-clockwise direction, whereas clockwise flight patterns were used to develop the NSM Expert (Fig. 1). Additionally, the takeoff point on Trajectory #2 was five degrees farther north than the training trajectories' takeoff points. These trajectory differences change the measurement and INS dynamics, and hence the hybrid performance. Trajectory #2 was designed this way intentionally, so that the NSM Expert System's adaptability could be determined.

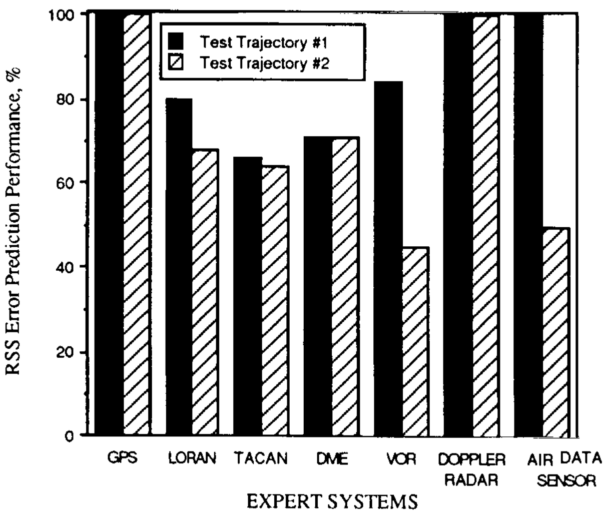
Single-, double-, and triple-station combination hybrids were simulated on each test trajectory for each of the DME, VOR, TACAN, and LORAN systems. The combinations were formed using four ground stations located as in Fig. 1 with respect to each other. Additionally, two-, three-, and four-satellite hybrids were simulated on the test trajectories, as were Doppler Radar and Air Data sensor hybrids. In total, sixty covariance simulations were performed for the two test trajectories.

Test Trajectory Data Preparation, Performance Metrics, and Results

The performance results for each of the sixty simulations were classified on each trajectory leg according to the scheme in Table I. The total number of matches was counted on each leg of each test trajectory for the seven navaid types studied. A match was declared between the actual and predicted RSS classification if and only if the RSS classification codes differed by one or less. For example, if the NSM Expert predicted an RSS classification code of 6 whereas the covariance results determined a performance of Class 7, a match was declared. A match would also have been declared if the actual performance was Class 5. Since the NSM Expert is only expected to *estimate* a hybrid's performance, it is allowed some room for error.

In total, the NSM Expert System was run four hundred and eighty-eight times in order to determine the number of matches for each system on the test trajectories. Figure 10 shows the NSM Expert's performance in predicting the RSS position error for each hybrid configuration. The *predictive performance metric* for each navaid is defined as the percentage of number of matches obtained from the total number of combinations tested for that navaid. The matches on all four trajectory legs are reflected in this figure.

The NSM Expert performed very well on the two test trajectories. Figure 10 shows that the NSM Expert correctly predicts the RSS position error better than 70% of the time on test Trajectory #1. The system required only the trajectory information and its knowledge of hybrid system performance to make these predictions. However, its predictive capability on test Trajectory #2 is slightly worse for the LORAN hybrids, considerably worse for the VOR and Air Data sensor hybrids, and identical for the remaining configurations. Hence, the results from Trajectory #2 suggest that additional investigation into trajectory effects on VOR's and Air Data Sensor's performance may be necessary.



The results in Fig. 10 are truly encouraging for designers of expert systems. We have shown that an expert system can be designed from data, and that good results are obtainable even from relatively small training sets. Recall that the total number of examples used to obtain the NSM decision trees was slightly less than one thousand.

CONCLUSIONS

The performances of seven navigation systems aiding a medium-accuracy INS were investigated using Kalman Filter covariance analyses. Hybrid performance decisions were based on the RSS position error history metric. A NSM Expert was designed from covariance simulation data using a systematic method comprised of the two statistical techniques, the Analysis of Variance (ANOVA) method and the ID3 algorithm.

ANOVA results show that statistically different position accuracies are obtained when different navaids are used, the number of radio navigation ground stations or GPS satellites used to aid the INS is varied, the aircraft's trajectory is varied, and the performance history is varied. By indicating that these four factors significantly affect the decision metric, an appropriate parameter framework was designed, and a simulation example base was created.

The example base was composed of over nine hundred training examples from nearly three hundred simulations. The example base was divided into seventeen smaller groups to enhance manageability. The ID3 algorithm then was used to determine the NSM Expert's classification "rules" in the form of decision trees. The performances of these decision trees were assessed on two arbitrary trajectories, by counting the number of times the rules correctly predicted the RSS position accuracy. These performance results then were presented using a predictive metric.

The ANOVA/ID3 method was very effective for the systematic development of the NSM Expert using simulation data. Results show that the NSM Expert can predict the RSS position accuracy between 65 and 100% of the time for a specified navaid configuration and aircraft trajectory. The test trajectories used to evaluate the system's performance show that the NSM Expert adapts to new situations and provides reasonable estimates of the expected hybrid performance. The system's good performance with relatively few examples clearly shows how the ID3 algorithm maximizes the information content contained in the example base. The performance results strongly suggest that operational systems can be designed from simulation or experimental data using the ANOVA/ID3 method for knowledge acquisition. The systematic nature of the method makes it a useful tool for expert system designers.

Other aerospace applications that are good candidates for the ANOVA/ID3 method are air combat pilot strategies from simulation or flight test data and air traffic control solutions to multi-configuration problems. The expert system design methodology also is pertinent to problems such as nuclear reactor control strategies, chemical process control strategies, automated highway driving, and robotics applications. In each case simulation or operational experiments may be executed for the systematic development of an expert system advisor.

ACKNOWLEDGEMENTS

This project was sponsored by the U.S. Army Research Office under Contract No. DAAG29-84-K-0048, and supported by NASA and the FAA under Grant No. NGL31-001-252.

REFERENCES

[1] G. Kahn *et al,* "MORE: An Intelligent Knowledge Acquisition Tool", in *Proceedings of the Ninth International Conference on Artificial Intelligence,* Los Angeles CA, 1985.

[2] F.G. Unger and R.S. Sindling, "Systems Tasks for Advanced Aircraft NavigationSystems", in *Computers in the Guidance and Control of Aerospace Vehicles,* AGARD-AG-158, February 1972.

[3] A. Gelb and A. Sutherland Jr., "Software Advances in Aided Inertial Navigation Systems", *Navigation,* The Institute of Navigation, Washington DC, Vol. 17, No.4, Winter 1970-71, pp 358-369.

[4] J. Richman and B. Friedland, "Design of Optimum Mixer-Filter for Aircraft Navigation Systems", in *Proceedings of the IEEE 1967 National Aerospace and Electronics Conference (NAECON) 1967,* pp.429-438.

[5] W. Zimmerman, "Optimum Integration of Aircraft Navigation Systems", in *IEEE Transactions on Aerospace and Electronics Systems,* Vol. AES-5, No.5, September 1969, pp. 737-747.

[6] J. W. Burrows, "Combined Inertial-ILS Aircraft Navigation Systems", *Journal of Aircraft,* Vol. 8, No. 6, June 1971, pp.439-443.

[7] D. B. Cox Jr., "Integration of GPS with Inertial Navigation Systems", in *Global Positioning System Vol. II*, papers published in *Navigation,* The Institute of Navigation, Washington DC, 1984, pp. 144-153.

[8] E. J. Rose *et al., A Cost/Performance Analysis of Hybrid Inertial/Externally Referenced Positioning/Orientation Systems*, Report ETL-R-086, U.S. Army Engineer Topographic Laboratories, September 1985.

[9] A.D. Pisaro and H.L. Jones, "An Expert System Approach to Adaptive Tactical Navigation", in *Proceedings of the First Conference on Artificial Intelligence Applications,* IEEE Computer Society, 1982, pp. 460-464.

[10] S. Berning, D. P. Glasson and J. L. Pomarede, "Knowledge Engineering for the Adaptive Tactical Navigator", in *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, May 1988, pp. 1266-1273.

[11] G. E. P. Box, W. G. Hunter and J. S. Hunter, *Statistics for Experiments: An Introduction to Design, Data Analysis and Model Building*, John Wiley & Sons, New York, 1978, Chapter 6.

[12] *Expert Ease Manual*, Human Edge Software, Intelligent Terminals Ltd., Palo Alto, CA, 1983.

[13] J. R. Quinlan, "Discovering Rules by Induction From Large Collections of Examples", in *Expert Systems in the Micro Electronic Age*, D. Michie, Editor, Edinburgh University Press, 1979, pp. 169-201.

[14] B. L. Belkin, *Cooperative Rule-Based Systems for Aircraft Navigation and Control*, M.S.E. Thesis, Report 1856T, Department of Mechanical and Aerospace Engineering, Princeton University, June 1989.

[15] P. S. Maybeck, *Stochastic Models, Estimation and Control,* Vol. 1, Academic Press, New York, 1979.

[16] R. F. Stengel, *Stochastic Optimal Control -- Theory and Application*, J. Wiley & Sons, New York, 1986.

[17] M. Kayton and W. R. Fried, *Avionics Navigation Systems*, John Wiley & Sons, New York, 1969.

DEVELOPING A PROLOG-BASED MODELING ENVIRONMENT

R. Lindell
Aerospace Corporation

(Paper not provided at publication date)

WRAPPING MATHEMATICAL TOOLS

C. Landauer
Aerospace Corporation

(Paper not provided at publication date)

INTELLIGENT COMPUTER-AIDED TRAINING

PRECEDING PAGE BLANK NOT FILMED

N91-20693

THE DEVELOPMENT OF EXPERTISE USING AN INTELLIGENT COMPUTER-AIDED TRAINING SYSTEM

Debra Steele Johnson
Department of Psychology
University of Houston
Houston, TX 77204

ABSTRACT

(The work presented here was completed for the NASA/ASEE Summer Faculty Fellowship Program in 1989 [Technical Report #185601].) An initial examination was conducted of an Intelligent Tutoring System (ITS) developed for use in industry. The ITS, developed by NASA/JSC, simulated a satellite deployment task. More specifically, the PD (Payload Assist Module Deployment)/ICAT (Intelligent Computer Aided Training) System simulated a nominal Payload Assist Module (PAM) deployment. The development of expertise on this task was examined using three Flight Dynamics Officer (FDO) candidates who had no previous experience with this task. The results indicated that performance improved rapidly until Trial 5, followed by more gradual improvements through Trial 12. The performance dimensions measured included performance speed, actions completed, errors help required, and display fields checked. Suggestions for further refining the software and for deciding when to expose trainees to more difficult task scenarios are discussed. Further, the results provide an initial demonstration of the effectiveness of the PD/ICAT system in training the nominal PAM deployment task and indicate the potential benefits of using ITS's for training other FDO tasks.

INTRODUCTION

Intelligent Tutoring Systems (ITS's) have been developed for a variety of tasks, ranging from geometry to LISP programming (Wenger, 1987). However, many of these systems have been used primarily for research purposes and have not been widely used in academic or industrial settings. An examination is needed of an ITS developed for use in an industrial setting. More specifically, an examination is needed of the development of expertise on an ITS in an industrial setting.

Background on PD/ICAT

Recently, an ITS was developed at NASA simulating the deployment of a specific type of satellite. Researchers at NASA/JSC developed the PD (Payload Assist Module Deployment)/ICAT (Intelligent Computer Aided Training) system (Loftin, 1987; Wang, Baffes, Loftin, & Hua, 1989). The task selected for this ITS was unique in that it required highly specialized skills and required extensive training using traditional OJT (On the Job Training) methods. The population (i.e., Flight Dynamics Officers [FDO's]) performing this task were also unique in that they tended to be well-educated and highly motivated. The PAM deployment task is one of many tasks (e.g., Ascent, Entry, Perigee Adjust, Rendezvous, IUS deployments) performed by FDO's working in the Mission Control Room. The training period for certifying a FDO ranges from two to four years. Due to the high costs and time required for training, researchers at NASA/JSC were charged with investigating tools to more quickly and economically train FDO's. The PAM deployment task was selected for ITS development in part because it was of moderate difficulty compared to other FDO tasks. In addition, PAM deployments were very common at that time, so training on this task was likely to be immediately useful to a FDO (although the frequency of PAM deployments has declined more recently). Moreover, the PAM deployment task had components common to several other FDO tasks, so training on this task was expected to transfer in part to performance on other FDO tasks.

The PD/ICAT system included a domain expert (i.e., an expert model), a trainee model, a training session manager, a scenario generator, and an user interface (Loftin, 1987). The domain expert contained information on how to perform the task. The task was described by a sequence of required and optional actions. However, it was necessary to build some flexibility into the sequence because several alternative sequences were equally acceptable for subsets of the actions. The knowledge type could be described as "flat procedural", that is, as requiring procedural knowledge without requiring subgoaling (VanLehn, 1988). Because the PAM deployment task was a highly procedural task, the domain expert was constructed as a set of procedures. To model the trainee, the system used an overlay model and a bug library (VanLehn, 1988). The system assumed the trainee model was similar to the expert model, but with some procedures missing. Further, the trainee model enabled the identification of incorrect procedures through the bug library. It is important to note that although the expert and trainee models were built as a set of procedures, extensive declarative knowledge was required to understand and perform those procedures. The training session manager interpreted the student's actions and reported the results in system (statement of action taken) messages or provided coaching in tutor (error, hint, or help) messages. Moreover, as recommended by other researchers (Burton & Brown, 1982; Reiser, Anderson, & Farrell, 1985), the training session manager provided feedback at each step in the action sequence and provided different levels of help or hints depending on the frequency of specific errors. Information from the training session manager was also incorporated into the student's performance record. Thus, the trainee model and training session manager together performed the major functions of student modelling: updating the level of student performance, providing information to the tutor, and recording student performance (Biegel, et al., 1988). The training scenario generator was used to expose the student to scenarios of varying difficulty. Lastly, the user interface enabled the student to interact with the system to obtain, enter, and/or manipulate information and complete actions.

Development of Expertise on the PD/ICAT System

The experts identified a total of 57 actions (38 required; 19 optional) to perform the PAM deployment task. These actions were performed in sequence although some subsets of actions could be performed in varying orders. In addition, the experts identified 83 display fields to check on 8 different displays. Some actions were performed more than once (e.g., anchoring an ephemeris); similarly, some of the displays were viewed more than once (e.g., the Checkout Monitor display). Performance improvement was defined in terms of increasing performance speed, completing task actions in sequence, requiring less help, and checking display fields identified as important by the experts. These performance dimensions provided a means for examining the development of expertise on the task. Other researchers (Anderson, 1985; Chi, Glaser, & Rees; Stevens, Collins, & Goldin, 1982) have similarly described the development of skill or expertise in terms of increasing performance speed and decreasing errors. More specifically, the declarative phase of skill acquisition involves acquiring knowledge about the task. Performance at this phase tends to be slow and error-prone. The knowledge compilation phase of skill acquisition involves using declarative knowledge to build procedures for performing the task. In this phase, performance speed increases and errors are reduced as productions are built and refined.

The purpose of the current project was to map the development of expertise on the PD/ICAT task. The data collected would provide an initial examination of how efficiently novices learned from the PD/ICAT system and enable recommendations for further refinements to the software. To accomplish this, the novices' performance on various dimensions was mapped across task trials and patterns of performance examined.

METHOD

Subjects and Procedure

Three novices performed 12 task trials on the PD/ICAT. The novices were FDO candidates. None had previous experience with Payload Assist Module (PAM) deployments. Experience with other integrated simulation tasks ranged from a minimum of 12 hours of observing IUS (Inertial Upper Stage) Deployments to a maximum of 48 hours of observing IUS Deployments plus more than 60 hours observing and participating in other integrated simulations (e.g., Deorbit Preparation, Entry, Ascent, Perigee Adjust, Rendezvous).

Each novice agreed to work 15-20 hours on the task in approximately 3-hour blocks spaced over a few weeks. However, due to work and other constraints, each novice had a different schedule of work sessions. Also, novices performed multiple task trials in a single work session after the initial task trials (i.e., after 3 to 5 trials, depending on the novice).

Novices were asked to read the section on PAM deployments in the Spin-Stabilized Deployment section of the Procedures Manual prior to coming to their first session. At the first session novices were shown an example of the screen display and told how to use the keyboard and the mouse to enter and manipulate task information. They were asked to "think out loud" as they performed the first task trial, that is, to describe what they were doing. In addition, the novices were invited to give their comments about the task interface and to ask questions as they performed the task. Their description of their actions, comments, and questions were tape recorded. All comments on the interface and questions about the task were noted by the researcher. However, only questions about the mechanics of the task were answered. No information was provided about which actions to perform at various points in the task. The novices were also told that their comments about the interface would be discussed with the task experts and the PD/ICAT programmers. Following each session, novices were shown a computer-generated feedback report describing their performance, their comments and questions were noted, and the next work session was scheduled. They were asked to "think out loud" again for Trials 3 and 9 (Trial 8 for one subject who was available for only 11 trials). On all other trials, the novices performed the task without having their comments tape recorded. Their comments and questions were noted by the researcher, usually at the end of the task trial.

The 12 task trials were completed in 5-6 work sessions. Following the last work session, the novices were asked to complete two short, paper and pencil tests. First, novices were asked to sort a list of all task actions into the proper sequence as quickly and accurately as possible. Second, novices were asked to identify information fields on screen displays as quickly and accurately as possible. Printed copies of each screen display were provided on which novices circled or checkmarked information fields they thought they were supposed to check during the

PAM deployment task. Two of the novices completed these tests 7 days after and one novice 12 days after their last work session. Finally, novices were debriefed and thanked for their participation.

Measures

Performance measures were collected by the computer during task performance. The performance measures collected for each trial were: trial time, number of actions completed, number of errors, number of help requests, and number of display fields checked. Trial time referred to the time required (in minutes) to complete a task trial. Number of actions completed referred to the number of actions (with or without errors) completed by the novice rather than by the Training Session Manager. (The PD/ICAT system was structured such that when the novices made three consecutive errors while attempting to complete an action, the Training Session Manager used the domain expert to complete the action.) Number of errors was the sum of three types of errors: the number of actions performed in an incorrect sequence, typographical errors (i.e., inputs the computer was unable to interpret), and optional (but recommended) actions which were not performed by the novice. Number of help requests was the sum of two types of help requests: the number of times novices requested more information from a tutor message following an error and the number of requests for explanations of the current or last step of the task.

Finally, number of display fields checked was the sum of the checks made on 8 unique screen displays, some viewed multiple times (see Table I). The maximum score was 83 display checks. Data were not available for one other display (Detailed Maneuver Table 1) because the computer did not correctly record the number of display fields checked. Viewing any display was an optional (but recommended) action. (The PD/ICAT system was structured such that configuring and viewing each display constituted two separate actions. A display could be configured without being viewed.) The recommended sequence and frequency of viewing different displays was determined by experts and incorporated into the PD/ICAT software. The Vector Comparison Display, however, was the only display not viewed as often as recommended by the experts. Rather than penalize the novices for failing to check display fields on a display they failed to view, an average score was calculated. The score for the Vector Comparison Display was calculated as the average number of display fields checked each time the display was viewed (e.g., the score was 5 if the novice viewed the display twice and checked 4 and 6 fields on the first and second viewings, respectively).

Additional performance measures were collected using the paper and pencil tests administered after the task trials. Three performance measures were collected on the sorting task. Sorting time referred to the time (in minutes) required to sort the sequence of actions. Unacceptable reversals referred to the number of actions sorted in incorrect sequences. Acceptable reversals referred to the number of actions sorted in a sequence regarded by the experts as an acceptable alternate sequence of actions. Two performance measures were collected from the display checking task. Checking time referred to the time (in minutes) required to check display fields on the 8 displays listed in Table I. Number of display checks recalled was the sum of the fields checked on these 8 displays.

Table I. Description of screen displays and display checks.

Display	# of Viewings	# of Display Field Checks
Vector Comparison*	3	7, 6, 6
Trajectory Digitals	1	2
Checkout Monitor	4	9, 9, 9, 9
Trajectory Profile Status**	2	7, 7
Detailed Maneuver Table 2	1	7
Weight Gain/Loss Table	1	3
Supersighter	1	9
FDO Deploy Comp	1	12

*An average score was calculated from the 3 viewing opportunities.
**Only the score for the 2nd viewing opportunity was used. Data was not correctly recorded by the computer for the 1st viewing opportunity.

RESULTS

To examine how efficiently the novices learned the PD/ICAT task, their data was plotted for each performance measure. As discussed below the data indicated rapid performance improvements until Trial 5 and more gradual further improvements through Trial 12. A logarithmic function was used to describe the data in each measure.

As shown in Figure 1, the trial time required to perform the task decreased rapidly until Trial 5. Further performance speed improvements were more gradual. In Trial 1 only one novice completed the task and required 195 minutes. The mean trial time was approximately 46 minutes by Trial 5 and decreased to approximately 26 minutes by Trial 12. The data was described by a logarithmic function ($Y = 233.95 * X^{-.83}$), accounting for 87% of the

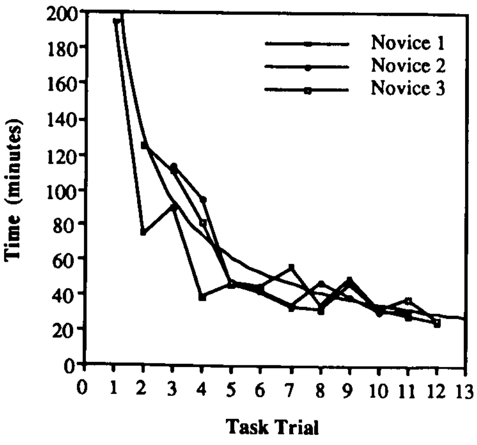


Figure 1. Performance speed in Trials 1 through 12.

variance. Interestingly, the novices who were unable to complete the task in the initial task trials demonstrated a performance pattern similar to that shown by Novice 1. Two novices failed to complete the task during the first 3-hour session, and 1 novice failed to complete the task until the third session. However, these novices demonstrated trial times similar to Novice 1 by Trial 5. Finally, the data indicates that the instruction to "think out loud" while performing the task slows performance speed. The time required to perform the task increased in Trial 3 for Novice 1, in Trial 8 for Novice 2, and in Trial 9 for Novices 1 and 3.

Number of actions completed also demonstrated rapid performance improvements until Trial 5 and then gradual further improvements. A logarithmic function ($Y = 34.49 * X^{.22}$) accounted for 63% of the variance (see Figure 2). In Trial 1, Novice 1 completed 43 actions out of the 57 possible actions. The remaining 14 actions were completed by the Training Session Manager, using the domain expert. Novices 2 and 3 completed only 28 and 26 actions, respectively. An additional 5 actions were completed by the Training Session Manager. Thus, Novice 1 completed 75% of the actions he attempted and Novices 2 and 3 completed 85% and 84% of the actions they attempted. However, one should note that Novices 2 and 3 completed or attempted to complete only 60% of the possible actions during Trial 1 while Novice 1 completed or attempted to complete all possible actions. The novices completed a mean of 52.33 actions in Trial 5 and a mean of 53.67 actions in Trial 12. Further, the novices completed at least 96% of the actions they attempted in Trial 5 and at least 98% in Trial 12. None of the novices attempted to complete more than 55 actions. Thus, novices chose not to perform at least 2 of the optional actions in every trial.

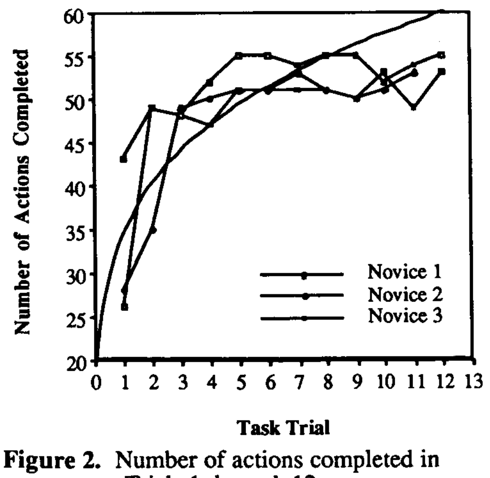


Figure 2. Number of actions completed in Trials 1 through 12.

Number of errors demonstrated a similar pattern of performance. A logarithmic function ($Y = 40.46 * X^{-1.00}$) accounted for 69% of the variance (see Figure 3). In Trial 1, the novices made a mean of 24 errors. Novice 1, however, made .54 errors/action attempted while Novices 2 and 3 made .58 and .87 errors/action attempted, respectively. By Trial 5, the novices made a mean of 4.33 errors and further reduced their errors to a mean of 3.5 by Trial 12. Thus, by Trial 5 the novices made a mean of only .08 errors/action attempted. By Trial 12, further performance improvements resulted in a mean of only .06 errors/action attempted.

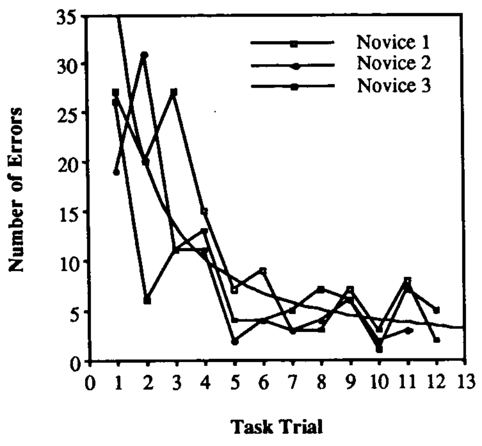


Figure 3. Number of errors made in Trials 1 through 12.

Similar to other performance measures, number of help requests demonstrated rapid reductions from Trial 1 to Trial 5, but there were few help requests following Trial 5. A logarithmic function ($Y = 46.44 * X^{-1.58}$) accounted for 91% of the variance (see Figure 4). (Note: The data in Figure 4 reflect a transformation of [X + 1] to enable a logarithmic function to be fit. Data reported in the text are in their original, untransformed units.) However, the novices showed much greater variability in their help requests than in other performance measures, especially in Trials 1 and 2. In Trial 1, the number of help requests ranged from 7 to 32 requests. The number of help requests varied even more in Trial 2, ranging from 1 to 49 requests. By Trial 3, however, the novices made similar numbers of requests with a mean of 7.33 requests.

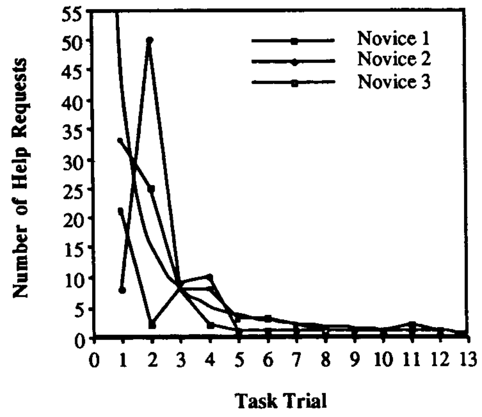


Figure 4. Number of help requests in Trials 1 through 12.

In Trial 5, the novices made a mean of .67 help requests and only one help request was made from Trial 8 through 12.

Finally, number of display fields checked demonstrated rapid performance improvements from Trial 1 to Trial 5 and more gradual improvements through Trial 12. A logarithmic function ($Y = 25.94 * X^{.50}$) accounted for 70% of the variance (see Figure 5). In Trial 1, the novices checked a mean of 10.33 display fields. However, only Novice 1 had the opportunity to check all 83 display fields because the other two novices did not complete the task in Trial 1. Thus, Novice 1 checked 13% of the appropriate display fields. Novice 2 checked 12% of the 34 display fields he viewed, and Novice 3 checked 59% of the 27 display fields he viewed. Although Novice 3 checked a higher percentage of display fields than the other novices, it is not clear that he understood which fields should be checked. He may have checked numerous fields because he was unsure which were important. The task software did not record checks of any display fields other than those identified as important by the experts. Thus, following Trial 1, the novices were instructed to check only those fields they considered important in each display. In Trial 5, the novices checked a mean of 69.22 fields which was 80% of the identified display fields. By Trial 12, the novices checked a mean of 79.89 fields, checking 96% of the identified fields.

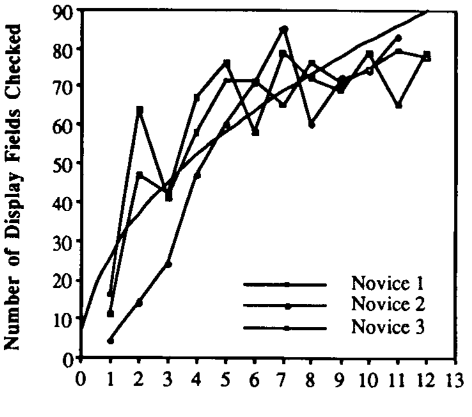


Figure 5. Number of display fields checked in Trials 1 through 12.

The results from the two paper and pencil tests were examined to determine whether the novices knew the correct sequence of actions in the task and whether they knew which display fields were important to check, as identified by the experts. The results of the sorting test indicated that Novices 2 and 3 required 17.08 and 17.92 minutes, respectively, to sort the task actions into the correct sequence. These two novices made 5 and 2 reversals, respectively, in how they sequenced the actions, but both reversals reflect alternate sequences regarded as acceptable by the experts. Novice 1 required 29.25 minutes to sort the task actions and made 4 acceptable and 3 unacceptable reversals. Of the 3 unacceptable reversals, one action was placed to soon, a second action too late, and the third action omitted from the task sequence. Thus, Novices 2 and 3 were able to correctly sort the task actions even after a 7-day delay. However, Novice 1 made 3 errors in sorting the task actions after a 12-day delay.

The results of the display checking task indicated that the novices required between 3.18 and 6.58 minutes to complete the task. They checked between 62 and 68 display fields, with a mean of 64.33. Of the total fields checked, between 40 and 51 (with a mean of 44.67) of the display fields were those identified as important by the experts. Thus, the novices checked 77% of the 58 identified display fields. However, the novices also checked between 17 and 22 (with a mean of 19.67) display fields not identified as important. This indicated that 31% of the fields the novices checked were not identified as important by the experts.

DISCUSSION

The results indicated that performance improved most rapidly from Trial 1 to Trial 5 on the PD/ICAT task. Additional task trials showed smaller, more gradual improvements. This suggests that the novices had developed effective procedures for performing the task by Trial 5. Additional task trials enabled the novices to refine these procedures, increasing performance speed and decreasing errors. If the goal is to train the novices to perform this specific task version as efficiently as possible, additional practice in Trials 6 through 12 may be warranted. However, the novices performed only the nominal PAM deployment task on the PD/ICAT. They also need to learn how to deal with problems that can occur during a PAM deployment, e.g., an OMS (Orbital Maneuvering Subsystem) propellant leak. So, given the smaller improvements following Trial 5, it may be reasonable after Trial 5 to expose the novices to more problematic PAM deployment scenarios.

Prior to making this decision, though, criteria should be identified for each performance dimension. That is, one needs to identify acceptable levels of performance in terms of time (in minutes) required to complete a task trial, number of completed actions (both required and optional), number of errors made, number of help requests, and number of display fields checked. These criteria, rather than a trial number, could then be used to determine when to expose a novice to a more difficult task scenario.

The results of the two tests administered after task performance indicated that the novices were able to recall the appropriate sequence of task actions a week after performing the last task trial, although there may be some decrements in recall for delays of more than a week. Similarly, the novices recalled 77% of the display fields to check after a week delay. However, decisions also need to be made here regarding 1) how many display fields should be recalled and 2) the potential benefits or costs of checking display fields not identified as important by the experts. In the nominal PAM deployment task the novices performed, no costs were associated with checking fields other than those identified. One needs to determine under what conditions it is acceptable and perhaps even desirable to check additional display fields. Experts may need to rank order the importance of checking different displays.

Finally, a few comments on the task interface are needed. These comments are based on comments and problems reported to the researcher by the novices. First, the novices experienced difficulty in beginning the task during Trial 1. All three novices were unsure what the first step should be. Consequently, they received multiple error messages and may have become frustrated. To alleviate this problem it may be appropriate to provide novices with additional information prior to performing Trial 1. This information could be in the form of task instructions, an example of the task sequence performed by the computer as the novice observes, or perhaps step by step help in completing the task sequence in the first task trial.

Second, the novices reported that some displays should be accessible at any point in the task. The PD/ICAT task as currently designed allows the novice to request displays only at specific points in the task. The novices' reports should be clarified with experts and modifications made to the software to either provide novices with greater access to displays or more explanation about why they should or should not need to view a display at a specific point in time.

Third, all three novices had difficulty interpreting the error messages provided. Further refinements of the PD/ICAT task should include improvements in the tutoring (i.e., error messages) provided.

Finally, more consideration needs to be given to the data collected from novices' task performance. Observing the novices performing the task indicated that they often attempted to perform actions out of sequence, primarily in the initial task trials. However, while the PD/ICAT software currently records whether an action has been completed and the number of errors associated with that action, no record is made of the specific sequence in which the actions were attempted. Further refinements to the software should enable the recording of sequencing information. Similarly, the current PD/ICAT software records only checks of identified display fields. Thus, a possible task strategy for a novice would be to check every field in a display to ensure that the machine recorded s/he had checked the important fields. A future enhancement of the software should include recording all display fields checked and perhaps providing information to the novice on why the identified fields are important to check.

CONCLUSIONS

Novices can efficiently learn to perform the PD/ICAT task which simulates a nominal PAM deployment. Additional work is needed to more clearly identify performance criteria and expand the PD/ICAT software to include more problematic PAM deployment scenarios. Finally, refinements are needed to improve the tutoring (error messages) provided and to assist the novice in performing the first task trial. The generally positive results of this project provide an initial demonstration of the effectiveness of the PD/ICAT software in teaching novices a nominal PAM deployment task and indicates the potential benefits of future refinements and expansions of the PD/ICAT software.

REFERENCES

Anderson, J. R. (1985). *Cognitive Psychology and its Implications (2nd Edition).* NY: W. H. Freeman.

Biegel, J. E., Interrante, L. D., Sargeant, J. M., Bagshaw, C. E., Dixon, C. M., Brooks, G. H., Sepulveda, J. A., & Lee, C. H. (1988). Input and instruction paradigms for an intelligent simulation training system. *Proceedings of the 1st Florida Artificial Intelligence Research Symposium (pp. 250-253).*

Burton, R. R. & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems.* NY: Academic Press, 79-88.

Chi, M. T. H., Glaser, R., & Rees, E. Expertise in problem solving. In R. J. Sternberg (Ed.), *Advances in the Psychology of Human Intelligence (Vol. 1).* Hillsdale, NJ: Erlbaum, 7-76.

Loftin, R. B. (1987). *A General Architecture for Intelligent Training Systems.* Final Report, NASA/ASEE Summer Faculty Fellowship Program, Johnson Space Center, Contract No. 44-001-800.

Reiser, B. J., Anderson, J. R., & Farrell, R. G. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. *Proceedings of the 9th International Joint Conference on Artificial Intelligence (Vol. 1, pp. 8-14).*

Stevens, A., Collins, A., & Goldin, S. E. (1982). Misconceptions in students' understandings. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems.* NY: Academic Press, 13-24.

VanLehn, K. (1988). Student modelling. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Erlbaum, 55-78.

Wang, L., Baffes, P., Loftin, R. B., & Hua, G. (1989). An intelligent training system for space shuttle flight controllers. *Proceedings of the 1989 Conference on Innovative Applications of Artificial Intelligence.*

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann.

N91-20694

RULE-BASED SIMULATION MODELS

JOSEPH L. NIETEN
KATHLEEN M. SERAPHINE

LOCKHEED SPACE OPERATIONS
TITUSVILLE, FLORIDA

ABSTRACT

This paper describes procedural modeling systems, rule-based modeling systems and a method for converting a procedural model to a rule-based model.

Simulation models are used to represent real-time engineering systems. A real-time system can be represented by a set of equations or functions connected so that they perform in the same manner as the actual system. When the real-time system is being modeled, the modeler will have to code the system's calculations and characteristics using some computer language or modeling tool.

Most modeling system languages are based on FORTRAN or some other procedural language. Therefore, they must be enhanced with a "reaction" capability. This reactive capability allows the model to perform only those calculations which are dependent on information that has changed. Even with this capability, a procedural model must look at every variable to determine if a calculation which depends on that variable should be performed.

Rule-based systems are reactive by definition. Once the engineering system has been decomposed into a set of calculations using only basic algebraic unary operations, a knowledge network of calculations and functions can be constructed. With the network in place, the rule-based system merely reacts to changes in the data. When a variable is changed, those calculations which depend on it become active. The rule-based system will continually execute, performing all dependent calculations appearing on the agenda.

The knowledge network required by a rule-based system can be generated by a knowledge acquisition tool or a source level compiler. The source level compiler would take an existing model source file, a syntax template, and a symbol table and generate the knowledge network. Thus, existing procedural models can be translated and executed by a rule-based system.

Simulation

Simulation is used in the real world to represent real systems, so that users can perform operations on a system that reacts the same as the real system without the cost or danger involved with the real system. Simulation systems allow an engineer to perform 'what-if' scenarios against their designs of real systems. Dangerous situations, expensive failures, and real life incidents can be recreated without risk using simulation. The newest use of simulation systems is in the training community. The training world has been incorporating simulation models into their training systems. The difference between a simulation system and a training system is an intelligent tutor in the training system, which interacts with the student and the simulation models to provide a high fidelity training session.

Simulation Models resemble mathematical models in that both have the same purpose and both utilize mathematical relationships to represent real systems. However, closed-form, analytic equations do not represent complex real systems without some enhancements. Simulation models use the black box approach. Each component of the real system is represented by a 'black box' with inputs and outputs. Further, each box is a complex combination of continuous and discrete functionality, which must be represented by some mathematical relationship while preserving the continuous nature of some of the calculations. This is accomplished by maintaining the current state of each variable used within the model, while performing the numerical analysis dictated by the functionality of the 'black box'. A great deal of temporal reasoning is required of the modeler during this decomposition process.

Real-time execution of simulation models is an important concept/issue. In order for a simulation model to be considered real-time, it must be at the exact same state as the real system at any given point in time. This means that both the computer hardware and the model executive must be capable of performing a tremendous number of calculations per second. Further, the model executive must minimize the number of calculations required, so that real-time performance becomes a direct factor of the computer hardware. This means the model executive must be reactive and only schedule those calculations which have been stimulated. A calculation is stimulated when a variable which provides input to that calculation is changed. Not every application requires a real-time model, some can function with a near-real-time model. This is a model which is in sync with the real system some acceptable percent of the time, or has no more than a certain delay when providing output.

Procedural Simulation Methods

Simulation models have been developed using a variety of generic languages such as FORTRAN, COBOL, PL/1, C, and Pascal. There are even models which have been developed using tools which are based on one of these languages. Those model based on the newer tools have the ability to represent continuous and discrete systems, and can schedule calculations to some degree. However, in general, artificial intelligence features in the standard commercial simulation languages is not yet available, mostly they still rely on a procedural approach to performing simulation operations.

A procedural model will execute in a linear manner with well defined entry and exit points. This means, that the model must perform all operations that are between the entry and exit points of a particular procedure. Normally, this would not have an impact, however, the simulation process may require other areas of the model to perform calculations based on the new information generated by the calculation that just finished. In addition, that calculation itself may actually have to be performed again. Now, if the computer is busy finishing one set of calculations, it cannot (not yet anyway) also begin to perform any other calculations in tandem. Even if the computer could perform these tasks, how is it going to know that these tasks need to be performed - unless they are scheduled.

There are some systems today which can schedule tasks or calculations for execution. These systems take a procedural model and segment it into smaller pieces, which can then each be executed from start to finish without any need to communicate with the other segments. These segments can also be managed by an executive which schedules them for execution based on some ranking algorithm. This is a highly efficient method for performing complex simulations, however, this modeling system still has performance considerations. Even so, there are times when the amount of data which is changing overwhelms the model executive and it falls behind, thus making it less than real-time. This is a common problem when the real system is very complex and/or very large.

While segmenting a procedural model is very effective, the segmentation does not go down to the lowest level of execution. By segmenting a procedural model into some number of calculations, the size of the segment can be limited. However, there are still some calculations within that segment which do not have to be performed. The only way to absolutely perform only those calculations which have been stimulated is to segment down to the lowest possible level. Once this segmentation is accomplished the final representation can be analyzed and optimized. This level of analysis and execution is going to be very difficult using a procedural methodology or tool.

Neural Simulation Methods

Neural Simulation Modeling represents an expansion of Neural Network and Rule-Based Programming techniques. Neural Simulation Models have two main components: A rule-based model executive and a neural network.

The model executive contains explicit rules which define how to interpret and perform operations on the information contained within the neural network. A basic knowledge of unary mathematic operations, as well as a working knowledge of the more complex functions which are used to represent physical systems mathematically, are the heart of the model executive. All function operations and variable information are imbedded within the neural network. So, the executive must take the function and variable information stored in the associated nodes and adjust the contents of the current node depending on that information. When a node is changed, the model executive traverses the network, performing any operations which are dependent on the updated node. By using a rule-based executive, this process can be totally reactive, which implies a higher degree of efficiency (in general) than procedural execution of models, since nodes will only be adjusted as they are stimulated. Thus, a stimulus driven environment is required for execution of the model executive.

Even though a stimulus driven environment is ideal for model execution, some controls must be added. Some calculations are self stimulating; that is they update a variable which also provides input for the calculation. This can trigger an infinite loop, since most rule-based systems rely on recency to schedule a rule for execution. This problem can be controlled by forcing those types of calculations to cycle in a two stroke fashion, similar to a two cycle lawn mower engine. In other words, those calculations which can trigger their own execution will not be able to do so until the next cycle.

The most difficult problem with implementing a rule-based model executive is timing and/or calculations dealing with a change over time. The impact of this problem can be avoided if some simple steps are taken. Those calculations which require a 'Delta T' must be time stamped. Whenever the calculation is executed, the time that the last execution occurred must be available to calculate the time elapsed. Since every system has some kind of internal clock or time elapsed register, this impact becomes an issue of machine cycles required to do the time calculations versus performance. Another possible impact occurs when a model tries to use a time delay for calculations. This impact appears to effect only those operations which are binary in nature. Therefore, this impact can be minimized with better modeling techniques.

The neural network uses nodes to store data and function information and uses connections to define input, output and directional characteristics. This type of representation allows for the most powerful use of the knowledge stored in its nodes. The mathematical equations which represent the modeled physical system have a sophisticated connection scheme, when they are all combined for the purposes of simulation activities. The most efficient way to represent some physical system would be to first identify the independent variables within the system. Then identify each level of abstraction which

C-6

relies on those independent variables, and continue this process until all levels of abstraction have been identified. The final result should be those variables which are considered 'output only'.

As a result of this process, all nodes in the network will be unique. This eliminates redundant calculations and operations. Once a network representation is in place, the network can be 'scanned' for closed operations. These are operations which can be calculated in a linear or procedural manner without impacting the efficiency of the network. In actuality, these operations can be replaced by another abstract function, which is in turn added to the model executive. This is not a trivial task, since the system may try to over simplify the network representation and inadvertently change the functional representation of the network. Therefore, a degree of intelligence is required when performing any network operations/optimizations.

CLIPS was chosen to be the platform for this development effort. CLIPS is versatile and portable; making the decision relatively easy. It is also a NASA product, so the product life was not an issue. CLIPS is not the perfect solution, at least not version 4.3. The activation algorithm could be optimized to improve real-time performance and there are not any intrinsic timing functions, which are required to do some real-time calculations against time. Even so, CLIPS is the best software for the job at hand.

CLIPS has a high potential in the distributed processing arena. Tasks which are queued on the agenda could be executed on any available processor, thus, increasing real-time performance. This would enable simulation performance and capabilities to be directly related to the hardware platform.

Neural Model's can be built using a Knowledge Acquisition tool or some other Graphical User Interface. This is a major improvement in the simulation arena. This also represents the ultimate situation, where older models can be converted to a newer technology and then maintained using a state of the art GUI. Another scenario could be that once the model is converted, you do not have to use a GUI. Instead, you could continue to build models with the old system and use the new compiler to debug or test the model before it is put into production on the simulation system. This alone could have a major impact on productivity.

Conversion Techniques for Simulation Models

Why would anyone want to convert simulation models? This is the one of the most frequently asked questions in the simulation community. Simulation systems are very complex by nature. They can also be dependent on some specific computer hardware for execution. As a result, these simulation systems are not portable to other computer hardware platforms. The technology base line has changed considerably in the last five years making the computer hardware developed during that time 'obsolete'. Hardware obsolescence is forcing the migration of simulation systems to more modern platforms. The user community has identified the need to consider portability when making decisions concerning the future of their simulation systems.

The decision one must make during model conversion is whether to convert the models themselves to a new and improved modeling system or to convert the compilers and executives used to generate and/or execute these models. Something to consider when making this decision is the composition of the job. The actual 'coding' of the model accounts for only 50% of the overall task. The remaining 50% of the work is spent doing the mathematical analysis of the physical system. In other words, at least 50% of the development time is spent building the knowledge about the physical system being modeled, into a mathematical system. The number of man-hours invested in this component of development can represent a large investment. Therefore, it makes sense to convert these models at the source code level in order to retain the investment made in building the mathematical representation of the physical system.

Source level conversion can be complex. However, the real trick to being able to do a conversion of this type is the method of building the new mathematical representations from the old formats. While developing a new compiler is not a trivial task, it is a lot cheaper than the other options and certainly better than rewriting several hundred thousand lines of code.

The source level compiler takes an existing model source file, a syntax template, and a symbol table and generates the neural network, which is used by the model executive to perform the simulation. The compiler uses a mapping function to transform the original syntax into a modified neural network. The network must then be optimized in order to be used efficiently. It must have all duplicate nodes or covers (groups of nodes) combined and/or eliminated. All nodes not explicitly listed as either inputs or outputs must be eliminated as well.

The model compiler has to have a degree of intelligence, so that it can identify when to stop performing optimizations. It also should have the capability to request clarification on a node's status. The compiler should also be able to isolate work-around techniques used by the programmer. Periodically, programmers will develop a method for accomplishing some high order function, which was not available through the standard functional syntax of the original modeling system. This 'work-around' should appear in the network as a pattern where it can be identified and dealt with. A work-around can either be transformed or deleted depending on the mapping function.

Another method of conversion would involve porting the development tools to a new platform. This will allow for the continuation of existing model source. However, translating the base language of the development tools into a new language will not necessarily improve any of the internal algorithms nor will it guarantee portability. Portability will always be a major issue. In order for any product to survive in today's rapidly advancing technical world, <u>It must be portable</u>. Productivity is also an issue, since the development tools may be outdated to begin with. Converting outdated tools would be equivalent to giving an older car a paint job, but not a new engine.

Conclusions

Neural (Rule-based) Simulation techniques are extremely powerful and even though there are some problems with the implementation of this technology, neural models can provide the high capacity data manipulation required by the most complex real-time models. This technology is worth further investigation.

References

1. J. W. Schmidt, "Introduction to Simulation,"Proceedings of the 1984 Winter Simulation Conference, Dallas, TX, Society for Computer Simulation, San Diego, CA, 1984

2. R. Shannon, "Artificial Intelligence and Simulation," Proceed ings of the 1984 Winter Simulation Conference, Dallas,TX, Society for Computer Simulation, San Diego, CA, 1984

3. W. M. Holmes, Artificial Intelligence and Simulation, Society for Computer Simulation, San Diego, CA, 1985.

4. D. E. Rumelhart, G. E. Hinton, and J. L. McClelland," A General Framework for Parallel Distributed Processing," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, VOL. I, MIT Press, Cambridge, MA, 1986

5. B. P. Ziegler, Theory of Modeling and Simulation, Wiley, New York, 1976

INTELLIGENT TUTORING SYSTEMS: BREAKING THE PRICE BARRIER

Dr. Wesley Regian
Air Force Human Resources Laboratory
Brooks Air Force Base

(Paper not provided at publication date)

INTELLIGENT TUTORING SYSTEMS: BREAKING THE PRICE BARRIER

J. L. Fleming
Air Force Human Resources Laboratory
Brooks Air Force Base

(Paper not provided at publication date)

N91-20695

ROSE GARDEN PROMISES OF INTELLIGENT TUTORING SYSTEMS: BLOSSOM OR THORN?

Valerie J. Shute
AFHRL/MOE
Brooks Air Force Base, Texas

ABSTRACT

Intelligent tutoring systems (ITS) have been in existence for over a decade now. However, few controlled evaluation studies have been conducted comparing the effectiveness of these systems to more traditional instruction methods. This paper examines two main promises of ITSs: (1) Engender more effective and efficient learning in relation to traditional formats, and (2) Reduce the range of learning outcome measures where a majority of individuals are elevated to high performance levels. Bloom (1984) has referred to these as the "two sigma problem" -- to achieve two standard deviation improvements with tutoring over traditional instruction methods. Four ITSs are discussed in relation to the two promises. These tutors have undergone systematic, controlled evaluations: a) The LISP tutor (Anderson Farrell & Sauers, 1984); b) Smithtown (Shute & Glaser, in press); c) Sherlock (Lesgold, Lajoie, Bunzo & Eggan, 1990); and d) The Pascal ITS (Bonar, Cunningham, Beatty & Weil, 1988). Results show that these four tutors *do* accelerate learning with no degradation in final outcome. Suggestions for improvements to the design and evaluation of ITSs are discussed.

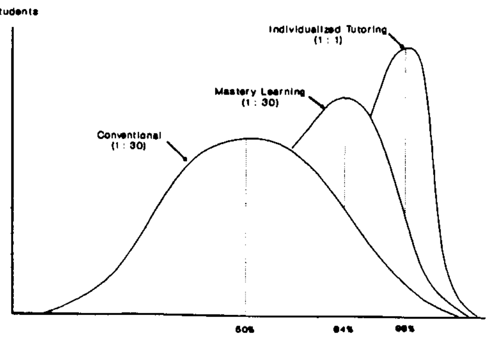
INTRODUCTION

Advances and innovations in the history of education have been scarce. Of the few instructional breakthroughs (e.g., Head Start program, "mastery learning"), none have conveyed more potential and excitement than the emergence of intelligent tutoring systems over a decade ago. For a long time, researchers have contended that individualized tutoring engenders the most effective and efficient learning for most people (e.g., Bloom, 1956, 1984; Burton & Brown, 1982; Carroll, 1963; Cohen, Kulik, & Kulik, 1982; Lewis, McArthur, Stasz & Zmuidzinas, 1990; Woolf, 1987). Intelligent tutoring systems (ITS) *epitomize* this principle of individualized instruction. Thus, by extension, the two main promises of ITSs are they can: (1) Engender more effective and efficient learning in relation to traditional formats, and (2) Reduce the range of learning outcome measures where a majority of individuals are elevated to high performance levels. These promises have been called the "two sigma problem" (Bloom, 1984). The goal is to achieve two standard deviation improvements with tutoring over traditional instruction methods.

For those of us concerned with teaching and learning, these promises of ITSs are profound. Unfortunately, although such systems have been in existence for over ten years now, their efficacy has been equivocal for several reasons: ITSs are often designed by seat-of-the-pants engineering, lacking principled design standards, and abounding in "intuition" underlying the implementation of system components (e.g., Koedinger & Anderson, 1990; Norman, 1989). Furthermore, systematic, controlled evaluations of ITSs are rare (Baker, 1990; Littman & Soloway, 1988). The few ITSs that actually have been evaluated in relation to other learning situations have shown evidence supporting the first promise (facilitating learning), but have shown little evidence supporting the second promise (reducing individual differences in outcome performance). I view this as encouraging, however, because new technologies usually do not fare well compared against proven methods (Baker, 1990).

Bloom (1984) identified problems associated with "proven", conventional teaching methods (e.g., a teacher presenting material in front of 30 people). He asserted that this format provides one of the least effective techniques for teaching and learning. As teaching becomes more focused and individualized, learning is enhanced. For example, when a teacher supplements a lecture with diagnostic tests to determine where students are having problems, then adjusts the lecture accordingly, this is called "mastery teaching". Students learning under this condition typically generate test results around the 84th percentile. Bloom further reported that students involved in "one-to-one tutoring", with human tutors, performed around the 98th percentile (2 standard deviation increase) as compared with traditionally-trained students (see Figure 1). These results were replicated four times with three different ages groups for two different domains. Bloom thus provides evidence that tutoring is one of the most effective educational delivery methods available.

Figure 1. Distributions for Different Learning Conditions
(adapted from Bloom, 1984)



This paper evaluates the two promises of one-to-one tutoring as embodied in four ITSs: a) The LISP tutor (Anderson, Farrell, & Sauers, 1984); b) Smithtown, an intelligent discovery world that teaches scientific inquiry skills in the context of microeconomics (Shute & Glaser, in press); c) Sherlock, a tutor for avionics troubleshooting (Lesgold, Lajoie, Bunzo, and Eggan, 1990); and d) The Pascal ITS, teaching Pascal programming skills (Bonar, Cunningham, Beatty, & Weil, 1988; Shute, in press). Results from these evaluations will be discussed in relation to the success criteria ("promises") as well as to ITS design issues.

FOUR EVALUATIONS

The LISP tutor. Anderson and his colleagues at Carnegie-Mellon University (Anderson, Farrell, & Sauers, 1984) developed a LISP tutor which provides students with a series of LISP programming exercises and tutorial assistance as needed during the solution process. In one evaluation study, Anderson, Boyle, and Reiser (1985) reported data from three groups of subjects: human-tutored, computer-tutored (LISP tutor) and traditional instruction (subjects solving problems on their own). The time to complete identical exercises were: 11.4, 15.0, and 26.5 hours, respectively. Furthermore, all groups performed equally well on the outcome tests of LISP knowledge. A second evaluation study (Anderson, Boyle & Reiser, 1985) compared two groups of subjects: students using the LISP tutor and students completing the exercises on their own. Both received the same lectures and reading materials. Findings showed that it took the group in the traditional instruction condition 30% *longer* to finish the exercises than the computer-tutored group. Furthermore, the computer-tutored group scored 43% *higher* on the final exam than the control group. So, in two different studies, the LISP tutor was apparently successful in promoting faster learning with no degradation in outcome performance compared to traditional instruction.

ORIGINAL PAGE IS
OF POOR QUALITY

In a third study using the LISP tutor to investigate individual differences in learning, Anderson (1990) found that when prior, related experience was held constant, two "meta-factors" emerged (i.e., factor analysis on factor scores). These two meta-factors, or basic learning abilities, included an *acquisition* factor and a *retention* factor. Not only did these two factors explain variance underlying tutor performance, they also significantly predicted performance on a paper-and-pencil midterm and final examination.

Smithtown. Shute & Glaser (in press) developed an ITS designed to improve an individual's scientific inquiry skills as well as provide a microworld environment for learning principles of basic microeconomics. In one study (Shute, Glaser & Raghavan, 1989), three groups of subjects were compared: a group interacting with Smithtown, an introductory economics classroom, and a control group. The curriculum was identical in both treatment groups (i.e., laws of supply and demand). Results showed that while all the three groups performed equivalently on the pretest battery (around 50% correct), the classroom and the Smithtown groups showed the same gains from pretest to posttest (26.4% and 25.2%, respectively), significantly outperforming the control group. Although the classroom group received more than twice as much exposure to the subject matter as did the Smithtown group (11 vs. 5 hours, respectively), the groups did not differ on their posttest scores. These findings are particularly interesting because the instructional focus of Smithtown was not on economic knowledge, *per se*, but rather on general scientific inquiry skills, such as hypothesis testing.

Another study conducted with Smithtown (Shute & Glaser, 1990) explored individual differences in learning and showed that scientific inquiry behaviors relating to a hypothesis generation and testing factor were significantly more predictive of successful learning in Smithtown than a standard measure of general intelligence. The five relevant indicators comprising this factor accounted for 42% of the criterion variance while a measure of general intelligence (composite of four tests) accounted for only 1% of the variance. These findings suggest that, in this tutor, individual differences in learning outcome are not simply a function of general intelligence. Rather, specific behaviors, presumably trainable, are predictive of outcome performance.

Sherlock. "Sherlock" is the name given to a tutor which provides a coached practice environment for an electronics troubleshooting task (Lesgold, Lajoie, Bunzo, and Eggan, 1990). The tutor teaches troubleshooting procedures for dealing with problems associated with an F-15 manual avionics test station. The curriculum consists of 34 troubleshooting scenarios with associated hints. A study was conducted evaluating Sherlock's effectiveness using 32 trainees from two separate Air Force bases (Nichols, Pokorny, Jones, Gott, & Alley, in press). Pre- and post-tutor assessment was done using verbal troubleshooting techniques as well as a paper-and-pencil test. Two groups of subjects per Air Force base were tested: (1) subjects receiving 20 hours of instruction on Sherlock, and (2) a control group receiving on-the-job training over the same period of time. Statistical analyses indicated that there were no differences between the treatment and the control groups on the pretest (means = 56.9 and 53.4, respectively). However, on the verbal posttest as well as the paper-and-pencil test, the treatment group (mean = 79.0) performed significantly better than the control group (mean = 58.9) and equivalent to experienced technicians having several *years* of on-the-job experience (mean = 82.2). The average gain score for the group using Sherlock was equivalent to almost four years of experience.

Pascal ITS. An intelligent programming tutor was developed to assist novice programmers in designing, testing, and implementing Pascal code (Bonar, Cunningham, Beatty, & Weil, 1988). The goal of this tutor is to promote conceptualization of programming constructs or "plans" using intermediate solutions. A study was conducted with 260 subjects who spent up to 30 hours learning from the Pascal ITS (see Shute, in press). Learning efficiency rates were estimated from the time it took subjects to complete the curriculum. This measure involved both speed and accuracy since subjects could not proceed to a subsequent problem until they were completely successful in the current one. To estimate learning outcome (i.e., the breadth and depth of knowledge and skills acquired), three criterion posttests were administered measuring retention, application and generalization of programming skills.

The Pascal curriculum embodied by the tutor was equivalent to about 1/2 semester of introductory Pascal (J. G. Bonar, personal communication, March 1990). That is, the curriculum equaled about 7 weeks or 21 hours of instruction time. Adding two hours per week for computer laboratory time (conservative estimate), the total time spent learning a half-semester of Pascal the traditional way would be at least 35 hours. In the study discussed above, subjects completed the tutor in considerably less time (i.e., mean = 12 hours, SD = 5 hours, normal distribution). So, on average, it would take about three times as long to learn the same Pascal material in a traditional classroom and laboratory environment as with this tutor (i.e., 35 vs. 12 hours).

While all subjects finished the ITS curriculum in less time compared to traditional instructional methods, there were large differences in learning rates found at the end of the tutor. For these subjects (having no prior Pascal experience), the maximum and minimum completion times were 29.2 and 2.8 hours, a range of more than 10:1. In addition, while all 260 subjects successfully solved the various programming problems in the tutor's curriculum, their learning outcome scores reflected differing degrees of achievement. The mean of the three criterion scores was 55.8% (SD = 19, normal distribution). The range from highest to lowest score was 96.7% to 17.3%, representing large between-subject variation at the conclusion of the tutor. In an attempt to account for these individual differences in outcome performance, Shute (in press) found that a measure of working memory capacity, specific problem solving abilities (i.e., problem identification and sequencing of elements) and some learning style measures (i.e., asking for hints and running programs) accounted for 68% of the outcome variance.

SUMMARY AND CONCLUSION

Intelligent tutoring systems have been around for over a decade now, so it is not unfair to ask: What is the verdict? Four ITSs have been discussed in this paper which have undergone systematic evaluations. The results of the evaluations, as a whole, were very encouraging. The common finding is that learning efficiency with ITSs was enhanced in relation to traditional instruction (e.g., LISP tutor, Smithtown, Sherlock, Pascal tutor). That is, learning rates were accelerated whereby students acquired the subject matter faster from various ITSs than from more traditional environments: (a) Subjects working with the LISP tutor learned the knowledge and skills in 1/3 to 2/3 the time it took a control group to learn the same material; (b) Subjects working with Smithtown learned the same material in 1/2 the time it took a classroom-instructed group; (c) Subjects working with Sherlock learned in 20 hours skills which were comparable to those possessed by technicians having almost 4 years experience; and (d) Subjects learning from the Pascal ITS acquired, in 1/3 the time, equivalent knowledge and skills as learned through traditional instruction.

For learning outcome measures, the LISP tutor yielded the same (or in one study, 43% better) criterion scores than a control group not using the tutor. Results from the Smithtown analysis showed that subjects learned the same material as a classroom group, despite the fact that the tutor focused on the instruction of scientific inquiry skills, not the subject matter. And the outcome data from subjects using Sherlock showed increases in scores comparable to an advanced group of subjects and significantly better than a control group. In all cases, individuals learned faster, and performed at least as well, with the ITSs as subjects learning from traditional environments.

The second promise, concerning a reduction in the *range* of outcome scores, was less straightforward to assess. While the outcome variance of the Smithtown data was fairly restricted (M=72.7; SD=10), posttest data from the Sherlock analysis showed a less restricted range in outcome scores (M=79; SD=17). And the results from the Pascal ITS study similarly showed a relatively large variability on the final performance measure (M=55.8; SD=19).

As stated earlier, Bloom (1984) reported that individualized tutoring resulted in a two standard deviation increase in outcome performance for the majority of learners (see Figure 1). He suggested that treatment-effect size be computed as follows: $(\text{Mean}_{\text{exper.}} - \text{Mean}_{\text{control}})/\text{SD}_{\text{control}}$. To illustrate, data from the Sherlock evaluation yields an effect size = (79.0 - 58.9)/19.7 = 1.02. This implies a 1 standard unit increase in performance above the control group of subjects (84th percentile). Although this represents a significant improvement of ITS over traditional instruction, it falls short of attaining "2 sigma" status.

The problem with finding evidence from the ITSs for a "reduction in range" may be due, in part, to the unreasonableness of the second promise. In a footnote to his article, Bloom reported, "The control class distributions were approximately normal, although the mastery learning and tutoring groups were highly skewed" (1984, p. 16). Skewness and kurtosis data were, unfortunately, not presented. It may be more reasonable to evaluate ITS success in terms of another criterion: the reduction in the *correlation* between incoming knowledge and skills and learning outcome. That is, for a tutor to be really effective, it should be able to compensate for (or remediate) incoming cognitive weaknesses, and reinforce strengths to maximize learning outcome. In terms of this criterion, Anderson (1990) reported two basic learning abilities (acquisition and retention factors) that were highly predictive of LISP outcome performance. A possible enhancement to the design of this system would include adapting to differences in learning abilities. For instance, on-line measures could be monitored for rates of acquisition and retention of the subject matter. Then subjects demonstrating deficits in either of these areas could receive compensatory instruction, as needed. In another study, Shute and Glaser (1990) identified certain inquiry skills that significantly predicted outcome performance for microeconomics. While this system did monitor inquiry skills, not enough adaptability was built into the design (i.e., it was created to be more exploratory so the "coach" intervened infrequently). A suggested system modification would include increasing intervention as needed, rather than only after a fixed number of "buggy" behaviors. Finally, findings from the Pascal tutor (Shute, in press) showed that learning outcome was strongly predicted by a working memory factor, two problem solving abilities, and some learning behaviors. Information about an individual's working memory capacity could be used to vary instruction, such as teaching smaller chunks of relevant knowledge for those with less working memory capacity. Moreover, this tutor could benefit from the inclusion of supplemental instruction on relevant problem solving skills (e.g., part-task training of sequencing skills). In summary, by restructuring curricular materials (i.e., adapting to individuals' needs in real-time), learning from tutors could become less dependent on aptitudes, thereby providing everyone with a "fair shake" at learning. Obviously this is an hypothesis that can be empirically verified with more research.

What else could bring ITSs closer to achieving these promises? A principled approach to the design and evaluation of ITSs would be very helpful. One such approach is exemplified by a taxonomy of learning skills, developed and currently in use for both basic and applied research at the Air Force Human Resources Laboratory (see Kyllonen & Shute, 1989). This taxonomy defines four interactive dimensions: subject matter, learning environment, desired knowledge outcome, and learner styles. It is believed that interactions among these dimensions influence outcome performance. For example, it is misleading to generalize that one type of learning environment (e.g., exploratory) is best for all persons. Rather, aptitude-treatment interactions (Cronbach & Snow, 1977) are believed to occur where certain learner characteristics (aptitudes and styles) are better suited to certain learning environments for optimal outcome performance. Controlled studies using the taxonomy are needed in order to test various combinations of interactive dimensions in ITS designs. Then controlled studies comparing ITSs versus traditional instruction are needed to calculate effect size measures and be related back to Bloom's "2 sigma problem". The taxonomy provides a useful metric for comparing and evaluating tutors.

In conclusion, the evaluation results are, overall, encouraging. This is rather surprising given the enormous differences among the four tutors in design structure as well as evaluation methods. The findings indicate these four tutors do accelerate learning with no degradation in final outcome. In addition to measuring the reduction in range of learning outcome (as indicated by the second promise), it was suggested that a supplemental criterion would be the attenuation of correlation between outcome score with incoming aptitude measures.

Obviously, further basic research is needed to add more "psychology" and control into ITS designs. Rather than continuing to build tutors randomly, a more efficient route to the goal of optimizing ITSs is to systematically alter the design of existing ones and evaluate the results of those changes in accordance with a principled approach (as is possible with the learning skills taxonomy). Many outstanding questions continue to beg for answers: What types of learners do better in what types of environments? Are certain domains better suited for specific instructional methods? When should feedback be provided, what should it say, and how is it best presented? How much learner control should be allowed? In conclusion, a principled approach to the design and evaluation of ITSs is badly needed before we can begin to obtain answers to these questions. Only then can we reassess the "verdict" of ITS success. Right now, ITSs are like rosebuds, as yet unopened, but foreshadowing beautiful flowers.

REFERENCES

Anderson, J.R., "Analysis of student performance with the LISP tutor," DIAGNOSTIC MONITORING OF SKILL AND KNOWLEDGE ACQUISITION, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

Anderson, J.R., Farrell, R., and Sauers, R, "Learning to program in LISP," COGNITIVE SCIENCE, Norwood, NJ, Vol. 8, 1984, pp. 87-129.

Anderson, J.R., Boyle, C. and Reiser, B., "Intelligent tutoring systems", SCIENCE, Vol. 228, 1985, pp. 456-462.

Baker, E. L., "Technology assessment: Policy and methodological issues," In H. L. Burns, J. Parlett, and C. Luckhardt (Eds.), INTELLIGENT TUTORING SYSTEMS: EVOLUTIONS IN DESIGN, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

Bloom, B.S. "Taxonomy of educational objectives: The classification of educational goals," In B. S. Bloom (Ed.), COGNITIVE DOMAIN, Handbook 1, McKay, New York, 1956.

Bloom, B.S., "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," EDUCATIONAL RESEARCHER, Vol. 13, No. 6, 1984, pp. 4-16.

Bonar, J., Cunningham, R., Beatty, P. and Weil, W., "Bridge: Intelligent tutoring system with intermediate representations," TECHNICAL REPORT, Learning Research & Development Center, University of Pittsburgh, Pittsburgh, PA, 1988.

Burton, R.R., and Brown, J.S., "An investigation of computer coaching for informal learning activities," In D. Sleeman and J.S. Brown (Eds.), INTELLIGENT TUTORING SYSTEMS, Academic Press, London, 1982.

Carroll, J., "A model of school learning," TEACHERS COLLEGE RECORD, Vol. 64, 1963, pp. 723-733.

Cohen, P.A., Kulik, J. and Kulik, C.C., "Educational outcomes of tutoring: A meta-analysis of findings," AMERICAN EDUCATIONAL RESEARCH JOURNAL, Vol. 19, No. 2, 1982, pp. 237-248.

Cronbach, L.J. & Snow, R.E., APTITUDES AND INSTRUCTIONAL METHODS: A HANDBOOK FOR RESEARCH ON INTERACTIONS, Irvington, New York, 1977.

Koedinger, K. R. and Anderson, J.R., "Theoretical and empirical motivations for the design of ANGLE: A new geometry learning environment," WORKING NOTES: AAAI SPRING SYMPOSIUM SERIES, Stanford University, Stanford, CA, 1990.

Kyllonen, P.C. and Shute, V.J., "A taxonomy of learning skills," In P.L. Ackerman, R.J. Sternberg, and R. Glaser (Eds.), LEARNING AND INDIVIDUAL DIFFERENCES, W.H. Freeman, New York, 1989, pp. 117-163.

Lesgold, A., Lajoie, S.P.., Bunzo, M., and Eggan, G., "A coached practice environment for an electronics troubleshooting job," In J. Larkin, R. Chabay and C. Sheftic (Eds.), COMPUTER-ASSISTED INSTRUCTION AND INTELLIGENT TUTORING SYSTEMS: ESTABLISHING COMMUNICATION AND COLLABORATION, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

Lewis, M.W., McArthur, D., Stasz, C., & Zmuidzinas, M., "Discovery-based tutoring in mathematics," WORKING NOTES: AAAI SPRING SYMPOSIUM SERIES, Stanford University, Stanford, CA, 1990.

Littman, D. and Soloway, E., "Evaluating ITSs: The cognitive science perspective," In M.C. Polson and J.J. Richardson, FOUNDATIONS OF INTELLIGENT TUTORING SYSTEMS, Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

Nichols, P., Pokorny, R., Jones, G., Gott, S.P., and Alley, W.E., "Evaluation of an avionics troubleshooting tutoring system," TECHNICAL REPORT, Air Force Human Resources Laboratory, Brooks AFB, TX, in press.

Norman, D.A., THE PSYCHOLOGY OF EVERYDAY THINGS, Basic Books, New York, NY, 1989.

Shute, V.J., "Who is likely to acquire programming skills?," JOURNAL OF EDUCATIONAL COMPUTING RESEARCH, Vol. 7, No. 1, in press.

Shute, V.J. and Glaser, R., "An intelligent tutoring system for exploring principles of economics," In R. E. Snow & D. Wiley (Eds.), IMPROVING INQUIRY IN SOCIAL SCIENCE: A VOLUME IN HONOR OF LEE J. CRONBACH, Lawrence Erlbaum Associates, Hillsdale, NJ, in press.

Shute, V.J., Glaser, R., and Raghavan, K., "Inference and discovery in an exploratory laboratory," In P.L. Ackerman, R.J. Sternberg, and R. Glaser (Eds.), LEARNING AND INDIVIDUAL DIFFERENCES, W.H. Freeman, New York, 1989, pp. 279-326.

Shute, V.J. and Glaser, R., "A large-scale evaluation of an intelligent discovery world: Smithtown," INTERACTIVE LEARNING ENVIRONMENTS, Norwood, NJ, Vol. 1, 1990, pp. 51-77.

Woolf, B. P., "A survey of intelligent tutoring systems," In Proceedings of the NORTHEAST ARTIFICIAL INTELLIGENCE CONSORTIUM, Blue Mountain Lake, NY, June, 1987.

REAL-TIME SYSTEM CONTROL

FLEXIBLE WING AIRCRAFT ROLL CONTROL USING FUZZY LOGIC

Stephen Chiu
Rockwell International Corporation

(Paper not provided at publication date)

REAL-TIME FUZZY LOGIC BASED TUNER FOR PID CONTROL

Sujeet Chiu
Rockwell International Corporation

(Paper not provided at publication date)

N91-20596

ASSESSING DAMPING UNCERTAINTY IN SPACE STRUCTURES WITH FUZZY SETS

Timothy J. Ross
Associate Professor
Department of Civil Engineering
University of New Mexico
Albuquerque, NM 87131

Timothy K. Hasselman
President
Engineering Mechanics Associates, Inc.
3820 Del Amo Blvd., Suite 318
Torrance, CA 90503

ABSTRACT

NASA has been interested in the development of methods for evaluating the predictive accuracy of structural dynamic models. This interest stems from the use of mathematical models in evaluating the structural integrity of all spacecraft prior to flight. Space structures are often too large and too weak to be tested fully assembled in a ground test laboratory. The predictive accuracy of a model depends on the nature and extent of its experimental verification. The further the test conditions depart from in-service conditions, the less accurate the model will be. Structural damping is known to be one source of uncertainty in models. This paper explores the uncertainty in damping to evaluate the accuracy of dynamic models. A simple mass-spring-dashpot system is used to illustrate a comparison among three methods for propagating uncertainty in structural dynamics models: the First Order Method, the Numerical Simulation Method and the Fuzzy Set Method. The fuzzy set method is shown to bound the range of possible responses and thus to provide a valuable limiting check on the First Order Method near resonant conditions. Fuzzy methods are a relatively inexpensive alternative to numerical simulation and they can be used to classify uncertain parameters into useful groupings.

INTRODUCTION

With the availability of high-speed digital computers and finite element modeling, it has become possible to model highly complex structural systems, such as the Space Shuttle, in great detail with tens of thousands of structural degrees of freedom (DOF). Structural dynamic models are greatly reduced, however, depending on their application. For example, dynamic analysis models may be reduced to thousands of DOF, test support models to hundreds of DOF, and control system models to tens of DOF. One of the chief concerns in model reduction is loss of accuracy, particularly in the very low-order models which represent the structural "plant" in controls applications. There are numerous other sources of inaccuracy as well, which can only be evaluated by testing the structure.

Testing for purposes of dynamic model verification usually involves a modal survey. While it used to take weeks or even months to conduct a modal survey using tuned sine dwell and analog data processing, the same number of modes can now be obtained in a matter of days using random vibration and digital data processing. Again, the digital computer has played a vital role in the development of this technology. Unlike the large mainframe computers used for analyzing finite element models, it is the minicomputers and microprocessors which over the past ten years have given impetus to the growth of experimental modal analysis. Experimentally derived modes are routinely compared with analytically predicted modes as a means of verifying an analytical model.

For the most part, analytical model verification is still performed intuitively, by trial and error. Experimental mode shapes and frequencies are compared with analytical predictions and the model is adjusted by hand in an effort to bring it into agreement with experimental data. Very often, the experimental modes are used directly in subsequent analysis rather than attempting the time-consuming (and sometimes unsuccessful) task of adjusting a model to match the data. There are many cases where the experimental modes cannot be used directly, however, as in the case of large space structures which are too large and/or too weak to be ground tested in their entirety. Models must then be relied upon to predict dynamic behavior, and the accuracy of these predictions is of major concern.

One of the problems confronting engineers today is, that while the tools for structural design, analysis and testing have individually matured over the past two decades, it is still not possible to predict with confidence how well a structure will perform in a given environment without direct experimental verification, i.e. without actually testing the operational configuration of the structure in an adequate simulation of that environment. In the case of large space structures, neither of these conditions can be met. The thermal, atmospheric and gravitational environment of space cannot as yet be adequately simulated in a ground test laboratory. Secondly, the fully assembled structures are too large to be tested in an earth-gravity environment except by either substructure or subscale testing. In many cases their configurations will change as transport vehicles dock and separate, as appendages are repositioned, as supplies are consumed, and as the structures grow or are otherwise altered to accomplish various space missions. Knowledge about the accuracy of a model is important information for the design of a control system; some inaccuracy can be tolerated by a robust controller, but there are tradeoffs between robustness and performance. The greater the uncertainty in the structural model, the poorer the control of the structure.

Two needs are therefore recognized: (1) to develop a means of evaluating the predictive accuracy of structural dynamic models when the structure cannot be tested under in-service conditions, and (2) to develop methods for enhancing the predictive accuracy of a model through a suitable program of ground testing. On-orbit testing will ultimately be required to obtain an accurate model of the "as-built" structure in space. However, the task of on-orbit identification will be greatly facilitated (perhaps only possible) by having first verified major portions of the structural model through ground testing.

This paper illustrates some of these ideas by focusing on the uncertainty of damping in structural dynamics models. The estimated modal damping matrix corresponds to the test modes. In general, a different modal damping matrix is obtained when the estimated damping matrix is transformed to the coordinate space of the analytical modes. The difference between the two provides one measure of damping uncertainty. The paper also demonstrates by numerical example three methods for propagating the uncertainties in modal mass, stiffness and damping forward through the model to evaluate the accuracy of response predictions, and backward to evaluate the uncertainties of model parameters. These three alternative methods for propagating uncertainty are the: First Order Method, Numerical Simulation Method and Fuzzy Set Method.

UNCERTAINTY IN DAMPING

The normal mode method is widely used for dynamic analysis of linear structures. By enabling the equations of motion to be written in terms of modal coordinates, solutions are more readily determined. Fortunately, structural damping tends to be small so that the classical undamped modes have a useful physical interpretation. It is common practice to introduce damping only after the equations have been transformed to modal coordinates. In this case, viscous damping is typically assumed and the modal damping matrix is taken to be diagonal. The elements along the diagonal are related to the percent of critical damping for each mode, while the rest of the matrix is neglected assuming that the modes are not coupled by damping forces in the structure. This assumption is valid whenever the modal frequencies are not closely spaced [1].

Although justification may be found for neglecting these terms in some analyses, there are times when this assumption is inappropriate. For example, when modal synthesis is employed to combine substructure characteristics in deriving the equations of motion for a complete structure, and linear viscous damping is taken to represent the dissipative mechanism of the structure, the full modal damping matrices are required for each substructure. Since the off-diagonal terms are likely to be of the same order as the diagonal ones, they too will influence the modal damping being computed for the complete structure.

The full modal damping matrix will also be useful in adjusting experimentally determined modal damping for structural models which must be revised to account for differences between earth and space environments. Such differences may include

- mass, stiffness and damping of suspension systems
- gravity loading
- thermal loading
- air damping

There are now several methods available for estimating the full modal damping matrix and for extracting complex modes from measured structural response [2-5].

METHODS FOR PROPAGATING UNCERTAINTY

There are a number of ways in which uncertainty can be propagated through a model. Theoretically, if probability distributions were known for the parameters of a model, and a functional relationship existed between the parameters and some desired response characteristic such as frequency response, then it would be possible to determine the probability distribution of that response characteristic. From a practical standpoint, however, this approach is not feasible. Probability distributions for the model parameters are rarely if ever available, and even if they were, the task of combining them to obtain the probability distributions of response would be exceedingly difficult. Fortunately, more practical alternatives do exist. Three are discussed in the following subsections, and then compared for a simple numerical example.

First Order Statistical Method

The First Order Statistical Method is perhaps the simplest, least expensive and most familiar approach. First order methods are based on linearization and are best suited to problems involving either linear or weakly nonlinear relationships among the parameters and input-output variables of the problem.

First order statistical methods are based on the principle of linear covariance propagation, or the linear transformation of covariance matrices from one set of variables to another. For example, suppose that r denotes a vector of random variables. These random variables might represent selected mass and stiffness parameters of a structural model which are not precisely known. The expected values of these random variables may be designated by the vector μ_r. The covariance matrix of the vector r is then

$$E[(r-\mu_r)(r-\mu_r)^T] = S_{rr} \tag{1}$$

Suppose further that the vector u represents a second set of random variables (e.g. eigenvalues and eigenvectors) related to r by u = f(r). The random variables, u, can be expressed in terms of the random variables, r, using a Taylor series expansion about the mean of u, denoted μ_u.

$$u = \mu_u + \partial u/\partial r\ (r - \mu_r) + \ldots\ldots.. \tag{2}$$

If the higher order terms are neglected, the covariance matrix of u is given by

$$E[(u-\mu_u)(u-\mu_u)^T] = E\{\partial u/\partial r\ (r - \mu_r)\ (r - \mu_r)^T\ \partial u/\partial r^T\} = S_{uu} \tag{3}$$

or

$$S_{uu} = T_{ur}\, E[(r-\mu_r)(r-\mu_r)^T]\, T_{ur}^T = T_{ur}\, S_{rr}\, T_{ur}^T \tag{4}$$

where E denotes the expectation operator, and

$$T_{ur} = \partial u/\partial r \tag{5}$$

In particular, the jkth element of the rectangular partial derivative matrix, T_{ur}, is the scalar quantity

$$(T_{ur})_{jk} = \partial u_j/\partial r_k \tag{6}$$

It is desirable to make the inverse transformation from u to r as well as the direct transformation from r to u. However, whereas one can express u as an explicit function of r, the converse is not true; one cannot write the functional relationship $r=f^{-1}(u)$ in explicit form. Instead, r and S_{rr} are obtained by statistical estimation [6-8]. The inverse transformation of the covariance matrix S_{uu} to S_{rr} is given by

$$S_{rr} = [(T_{ur})^T\, S_{uu}^{-1}\, T_{ur}]^{-1} \tag{7}$$

In Equation (7) the dimension of u must be greater than or equal to that of r.

Equation (7) is useful in the evaluation of predictive accuracy. A method is available to derive S_{uu} from sets of predicted and measured modal data whenever u represents modal frequencies and displacements. From this information it is possible to obtain by direct transformation (Equation 4) the covariance matrices of frequency response, impulse response or other measures of performance which are dependent on these. It is also possible to obtain the corresponding covariance matrix of parameter estimates by the inverse transformation (Equation 7).

Numerical Simulation

Numerical simulation is conceptually the simplest method for propagating random uncertainty through a model. The model may be linear or nonlinear, and the random parameters of the model may be assigned any desired distribution. Unlike linear covariance propagation where only the first two central moments of the parameter distributions are propagated, the entire distributions are propagated in numerical simulation. The chief disadvantage is the computational effort required.

In numerical simulation (or Monte Carlo simulation), parameter values are selected at random, and the model is exercised to compute the response quantities of interest. The desired parameter distributions are obtained by first using a random number generator to generate a sequence of numbers uniformly distributed between zero and one. The resulting sequence of numbers is used in the simulation. Because of the usual large number of calculations required for accuracy, this type of numerical simulation is not practical for large structural dynamics models. It is useful, however, for treating isolated nonlinearities, and for applications involving simple models.

Fuzzy Set Method

Fuzzy sets offer an alternative to random variables for representing uncertainty. Numerous works explaining fuzzy sets are available in the literature [9]. Whereas the uncertainty of a random variable is measured in terms of probability, the uncertainty of a fuzzy set can be measured in terms of possibility. Probability implies random uncertainty; however, possibility can be used to measure non-random uncertainty. The degree of uncertainty in a fuzzy set is defined by its membership value. The membership of a fuzzy set measures the level of possibility and ranges from zero to one. The degree of membership in a set can be thought of as a measure of the "belongingness" of a particular variable to that set. Fuzzy sets are used quite often to describe "linguistic" variables (such as light, moderate, heavy, etc.) where the variable can have a vague, or fuzzy meaning. Unlike probability density functions which define the relative frequency of occurrence of a random variable as a function of the values which the random variable may assume, the membership function defines the range of possibility of a fuzzy number as a function of membership. In the case of a triangular membership function where the vertex has a membership of unity, the value of the fuzzy number corresponding to the vertex is interpreted as the deterministic value.

It is important to keep in mind that the concepts of a density function and a membership function are different. A density function is based on probability theory which in turn is postulated from "crisp set" mathematics. A crisp set merely defines the sample space of a random variable; the variable is either in the sample space (membership = 1) or it is not (membership = 0). A fuzzy set differs from a crisp set (sample space) by allowing for vagueness in the prescription of the boundaries of the sample space. It is also noted that crisp sets are special subsets of fuzzy sets, and that probability theory is a special subset of possibility theory.

With this distinction in mind, one can attempt to relate the membership function of a fuzzy set to the probability density function of a random variable. This will be shown to be a useful relationship in the sense that it provides a means of bounding the uncertainty of response predictions, particularly when structural response is a highly nonlinear function of the uncertain model parameters. In this situation, first-order methods tend to be unreliable, and simulation methods too costly.

The propagation of uncertainties using fuzzy sets involves computations with interval variables and functions [9]. For example, a variable, x, could have as its value a or b or 3.5, etc. which are real numbers. Similarly, an interval variable, denoted by X, will have as its value [a,b]. All arithmetic operations on interval numbers can be applied to interval variables. A function of the interval variable X = [a,b] can be defined by

$$Y = f(x) = \{f(x) \mid x \in X\} = \{f(x) \mid x \in [a,b]\} \qquad (8)$$

whose value usually would be an interval number. When f(x) is continuous and monotonic on X = [a,b], Y can simply be obtained by

$$Y = \{\min [f(a), f(b)], \max [f(a), f(b)]\} \qquad (9)$$

The Vertex Method can be used to propagate uncertainties whenever Y is a function of many interval variables [10]. When $Y = f(X_1, X_2, \ldots, X_n)$ is continuous in the n-dimensional convex region, and no extreme point exists in this region (including the boundaries), then the value of the interval function can be obtained by

$$Y=f(X_1, X_2, \ldots, X_n) = \{\min_j [f(c_j)], \max_j [f(c_j)]\};\ j=1, n \qquad (10)$$

where $c_j = (X_{1j}, X_{2j}, \ldots\ldots X_{nj})$ represents the coordinates of the j^{th} vertex in n-dimensional space.

Comparison of Methods by Numerical Example

The amplitude and phase of the complex frequency response function (FRF) are important characteristics of actuator-to-sensor transfer functions for purposes of control-structure interaction. As a means of helping to evaluate the relative merits of the three alternative methods for propagating uncertainty presented in this section, numerical examples are presented for the amplitude of a complex FRF [11].

The equation of motion for a single-degree-of-freedom system subjected to a harmonic disturbing force f is given by,

$$m(d^2x/dt^2) + c(dx/dt) + kx = f(t) \qquad (11)$$

where m is the mass, c the damping coefficient, k the stiffness, x the displacement, and t is time. The amplitude of the complex FRF is given by the familiar formula

$$A(\Omega) = [(k - m\Omega^2)^2 + (c\Omega)^2]^{-1/2} \qquad (12)$$

where Ω is the frequency content of the force f. If it is assumed that m, c and k are all random variables with the probability density functions shown in Figure 1, where m_o, c_o and k_o correspond, respectively, to the nominal or mean value of m, c and k, one can write Equation (12) in the dimensionless form

$$A(m', c', k') = \{k' - m'\,\Omega'^2)^2 + 4\,\zeta_o^2\, c'^2\,\Omega'^2]\}^{-1/2} \qquad (13)$$

where: $m' = m/m_o$, $c' = c/c_o$, $k' = k/k_o$, $\Omega' = \Omega/\omega_o$

and where: $\omega_o = \sqrt{k_o/m_o}$; $\zeta_o = c_o/(2\sqrt{k_o m_o})$

Despite the simplicity in appearance of Equation (13), an analytic closed-form expression for the derived distribution of $A(\Omega)$ given the distributions of m, c and k is extremely difficult to obtain. Consequently, numerical methods are sought.

To apply the First Order Method, one must first differentiate Equation (13) with respect to m', c' and k'. These derivatives, $\partial A/\partial m'$, $\partial A/\partial c'$, $\partial A/\partial k'$ are quite complicated and have been documented elsewhere [12]. Then one must derive the mean and standard derivation of each of the normalized density functions in Figure 1. The mean in each case is simply unity. For the present example [11], the standard deviations for m', c' and k' (see Figure 1) are

$$\sigma_{m'}^2 = 0.0204, \quad \sigma_{c'}^2 = 0.2458, \quad \sigma_{k'}^2 = 0.0612$$

Finally, if lognormal distributions are assumed for the three uncertain parameters, the distribution functions for various frequency ratios (Ω') shown in Figure 2 are obtained.

These are approximate distributions, which should be good as long as the excitation frequency is not near resonance. In Figure 2, therefore, the plot for Ω' =1 may not be a good approximation. To verify this assumption, Monte Carlo simulations were run for the same four cases. The results of these simulations based on a sample size of 10,000, i.e., 10,000 evaluations of Equation (13), are superposed on the previously derived lognormal density functions in Figure 2 for comparison. As expected, the first-order approximations are valid for the three off-resonant cases.

If the structural parameters are estimated by triangular fuzzy sets which are similar in shape to the density functions given in Figure 1, the uncertainty in m, c and k (the base of the triangles) are the same in magnitude, but the peak membership at m_0, c_0 and k_0 are normalized to unity to provide for normal membership functions [9]. This can be done because in fuzzy sets, the area under the membership function does not have to be unity as is required for a probability density function. The processing required by Equation (10) is carried out using the Vertex method as previously described.

The derived fuzzy membership functions of FRF amplitude for the four excitation frequencies are shown in Figure 3. The curves in Figure 3 are similar to their counterparts in Figure 2, both in the spread and the frequency at which maximum amplification occurs. Comparison of the absolute density values (ordinates) in Figure 2 with the membership values (ordinates) in Figure 3 is not meaningful because of the theoretical differences between membership functions and density functions [9].

As a final example, all three uncertainty propagation methods are applied to a case where c'= 0.025 and Ω'= 0.975. This frequency ratio corresponds to the lower half power point of the frequency response function. The uncertainty in damping is assumed to be negligible for purposes of demonstration. At this frequency, the sensitivity of FRF amplitude to mass and stiffness is greatest, so that the first-order approximation should be at its worst. The results are plotted in Figure 4. Figure 4a shows the comparison between the First Order Method and Monte Carlo Simulation. Figure 4b shows the corresponding membership function. The unusually shaped distribution produced by the Monte Carlo Simulation is evidently the result of the (sightly) rounded peak of the FRF at an amplification of 20, which allows sampled amplitudes to "collect" in this narrow frequency band.

The example in Figure 4 illustrates how the first order method can yield unrealistically high response levels when $A(\Omega)$ is evaluated near resonance. It represents a deliberate attempt to force such a result, and is admittedly a pathological case. In reality, there will be damping uncertainty which will tend to extend the upper tail of the actual distribution, making the first order method a better approximation. However, in general, there is no guarantee that this will happen; the fuzzy set method therefore serves as a limiting check on the first-order method.

When using the vertex method, the number of required FRF calculations is given by:

$$n = 2N_a N_f N_r (2^{N_p}) \qquad (14)$$

where N_a = number of alpha cuts
N_f = number of frequencies
N_r = number of FRF's
N_p = number of uncertain parameters

This number, n, can become very large as N_p becomes large. Since the basic uncertain parameters in the present analysis are modal mass and stiffness parameters, N_p depends on the number of modes represented in the generic uncertainty model. For for a system with only 10 modes, N_p can be shown to be equal to 110 and n from Equation (14) is on the order of 10^{33}! However, since the fuzzy set method proves most helpful near resonance, only a few of the uncertain modal parameters should be important in these cases. A method to identify which of the modal parameters are important near resonance would be most desirable.

FUZZY CLASSIFICATION METHOD

An approach involving fuzzy classification [13, 14] is being explored to identify the most significant modal parameters that affect the FRF near resonance in multi-degree-of-freedom systems. In this approach a finite data set, $X = \{x_1, x_2, ...x_n\}$ is defined where each data set corresponds to an uncertain modal parameter. Each data set can be characterized by one or more features. The present analysis is looking at two features for each data set: (i) the coefficient of variation of the uncertain parameter which is obtained from the corresponding diagonal term of the covariance matrix (e.g. S_{rr}) of modal mass and stiffness parameters, and (ii) the sensitivity of the desired response quantity to that parameter (e.g. T_{ur}). The fuzzy classification method partitions these n data sets into c classes where c << n. In this case, the classes or groups might be a group of "important" parameters, a group of "unimportant" parameters, a group of "moderately important" parameters, and so forth, where the fuzzy membership value in a class/group could be a measure of its "importance"; i.e. a membership value of 1 would be important and a value of 0 would be unimportant.

As an example, suppose n = 6 and c = 2. The six data sets might correspond to the six independent modal mass and stiffness parameters of a 2-mode covariance matrix [15]. Define $class_1$ as the important parameters and $class_2$ as the unimportant parameters. The fuzzy classification algorithm begins by making arbitrary "crisp" class assignments for each of the data sets, as

	x_1	x_2	x_3	x_4	x_5	x_6
$class_1$:	1	0	0	1	0	0
$class_2$:	0	1	1	0	1	1

If one were interested in the response at some point on a structure near the first resonant frequency, and modes 1 and 2 were well separated, then one would expect only the diagonal modal mass and stiffness terms associated with the first mode (i.e., data sets x_1 and x_4) to be significant.

Details of the fuzzy algorithm have been described elsewhere [13], but the resulting fuzzy partition might take the following form:

	x_1	x_2	x_3	x_4	x_5	x_6
class$_1$:	.91	.08	.13	.95	.11	.07
class$_2$:	.09	.92	.87	.05	.89	.93

Each column of the fuzzy partition matrix above (denoted as the U matrix) defines the membership of a given data set (uncertain parameter) in each of the two classes. The columns must sum to unity regardless of whether U is a fuzzy or crisp partition matrix. In situations where the membership values are not all close to zero or one, additional classes (i.e., $c > 2$) might be assumed and another classification analysis conducted on the data sets.

The fuzzy classification method may be used directly to select a reduced parameter set for subsequent use in the fuzzy vertex method, or it may be used to construct a fuzzy relation which establishes the degree of relationship to which data set x_i and x_j are related. One such fuzzy relation, R, results from the computation,

$$R = (U{*}U^T)/n \tag{15}$$

where U is a fuzzy partition matrix segregating n data sets into c classes, and the operation * is algebraic [16]. This relation, R, gives a measure of the relative membership of the data clusters to individual classes. This relation has some special properties: it is always symmetric, the sum of its entries is unity, and the measure of "misclassification" is computed by subtracting the trace of the matrix R from unity. The diagonal elements of R give a measure of the total allocation of membership within a class and the off-diagonal elements yield a measure of the membership allocation between pairs of classes.

CONCLUSIONS

Damping is understood to be a major source of uncertainty in structural analysis. Of particular concern is the fact that damping has heretofore been unpredictable in complex structures; it must be determined experimentally for a prototype structure in an environment similar to that in which response must be predicted. In the case of large space structures, it will be impossible to measure damping directly because of physical limitations on ground testing, and because of the differences (atmospheric, thermal and gravitational) between the earth and space environments. Methods to accurately account for damping uncertainty will significantly improve plant models and afford opportunities for more accurate controllers of motion.

A comparison was made of three alternative methods for propagating uncertainty: the First Order Method, the Numerical Simulation Method and the Fuzzy Set Method. A single-degree-of-freedom mass-spring-dashpot system was selected for this purpose. Triangular probability density functions were defined for the mass, stiffness and damping parameters (m, k and c). Frequency response function (FRF) amplitude for displacement response to a force input was computed for the nominal damping ratio of 2.5%.

It was found that the Fuzzy Set Method bounds the range of possible responses and it provides a valuable limiting check on the First Order Method. The Fuzzy Set Method is a relatively inexpensive alternative to numerical simulation for propagating parameter uncertainty in complex models, whereas numerical simulation becomes prohibitively expensive.

The fuzzy classification method reveals clusters in the data in a multi-dimensional feature space. This automated procedure does not produce labels, vis a viz "important" or "unimportant". These labels have been assigned in a very arbitrary sense considering our limited understanding of damping. The utility in these methods for this problem would be in the area of experimental planning and in numerical code predictive accuracy. The notions of clustering of data in some feature space gives a clue of not only what kind of measurements to take, but maybe where to take them and whether redundant measurements are warranted. These methods eventually can act as a confirmation of behavior once specific sets of conditions are known to lead to some well understood response pattern.

ACKNOWLEDGEMENTS

This work was performed under the sponsorship of the National Aeronautics and Space Administration's Jet Propulsion Laboratory (Contract NAS7-1020), through a subcontract with Engineering Mechanics Associates (EMA), Inc. Appreciation is expressed to Jon Chrostowski of EMA for his efforts in many of the computations involved in this paper.

REFERENCES

1. Hasselman, T.K., "Modal Coupling in Lightly Damped Structures," AIAA JOURNAL, Vol. 14, No. 11, November 1976.

2. Hasselman, T.K., "Method for Constructing a Full Modal Damping Matrix from Experimental Measurements", AIAA JOURNAL, Vol. 10, No. 4, April 1972, pp. 526-527.

3. Juang, J.N. and Pappa, R.S., "An Eigensystem Realization Algorithm (ERA) for Modal Parameter Identification and Model Reduction", JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS, Vol. 8, No. 5, 1985, pp. 620-627.

4. Ibrahim, S.R., and Mikulcik, E.C., "A Method for the Direct Identification of Vibration Parameters from the Free Response," SHOCK AND VIBRATION BULLETIN, No. 47, Pt. 4, 1977, pp. 183-198.

5. Coppolino, R.N., "A Simultaneous Frequency Domain Technique for Estimation of Modal Parameters from Measured Data," SAE PAPER No. 811046, Aerospace Congress and Exposition, Anaheim, CA, October, 1981.

6. Eykhoff, P., SYSTEM IDENTIFICATION: PARAMETER AND STATE ESTIMATION, John Wiley and Sons, New York, NY, 1974.

7. Jazwinski, A. H., "Stochastic Processes and Filtering Theory", Vol. 64, MATHEMATICS IN SCIENCE AND ENGINEERING, Academic Press, New York, NY, 1970.

8. Hasselman,T.K.,"Structural Uncertainty in Dynamic Analysis," SAE PAPER No. 811049, Transactions, Society of Automotive Engineers, Inc., 1981.

9. Klir, G. and Folger, T., FUZZY SETS, UNCERTAINTY, AND INFORMATION, Prentice Hall, Englewood Cliffs, NJ, 1988.

10. Dong, W., and Shah, H., "Vertex Method for Computing Functions of Fuzzy Variables", JOURNAL OF FUZZY SETS AND SYSTEMS, Vol. 24, 1987, pp. 65-78.

11. Hasselman, T.K. and Chrostowski, J. D., "Evaluation of Predictive Accuracy in Structural Dynamics Models", Seventh International Modal Analysis Conference, Las Vegas, NV, February, Vol. 1, 1989, pp. 360-366.

12. Ross, T.J., Hasselman, T. K. and Chrostowski, J.D., "Damping Uncertainty in Structural Dynamics Models Using the Fuzzy Set Method", in review, 1990.

13. Bezdek, J. C., PATTERN RECOGNITION WITH FUZZY OBJECTIVE FUNCTION ALGORITHMS, Plenum Press, New York, NY, 1981.

14. Bezdek, J., Grimball, N., Carson, J., and Ross, T., "Structural Failure Determination with Fuzzy Sets," JOURNAL OF CIVIL ENGINEERING SYSTEMS, Vol. 3, June 1986, pp. 82-92.

15. Hasselman, T. K., "Methods for Evaluating the Predictive Accuracy of Structural Dynamics Models", REPORT No. TR-89-1152-4, NASA Jet Propulsion Laboratory, Pasadena, CA, June, 1990.

16. Bezdek, J. C. (1974) "Numerical Taxonomy with Fuzzy Sets", JOURNAL OF MATHEMATICAL BIOLOGY, Springer Verlag, Austria, Vol. 1, pp. 57-71.

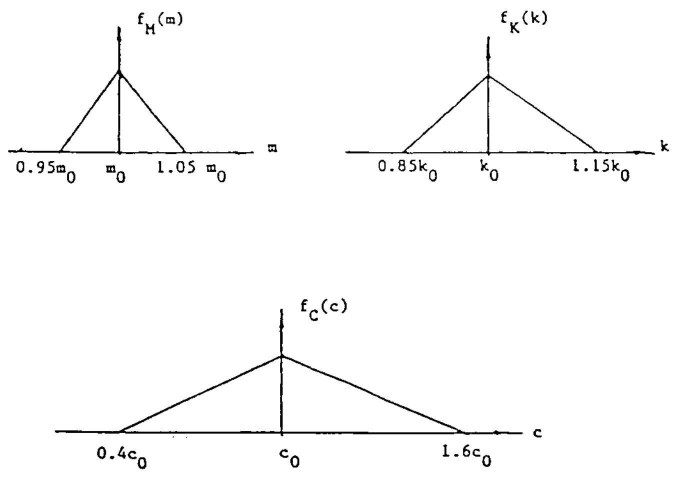


Figure 1. Triangular Forms for Uncertainties in m, k, and c (from [11])

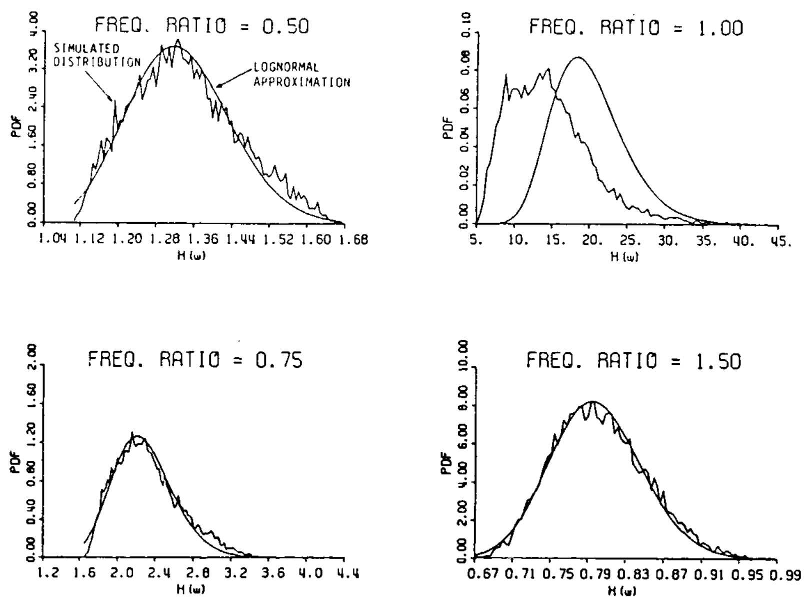


Figure 2. Comparison of First Order Method (lognormal) with Simulation Method (from [11])

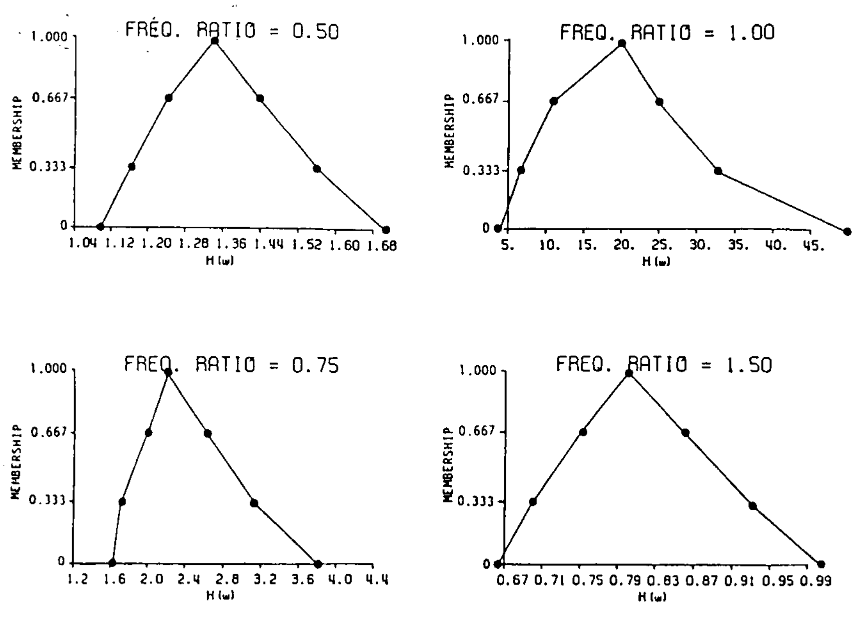
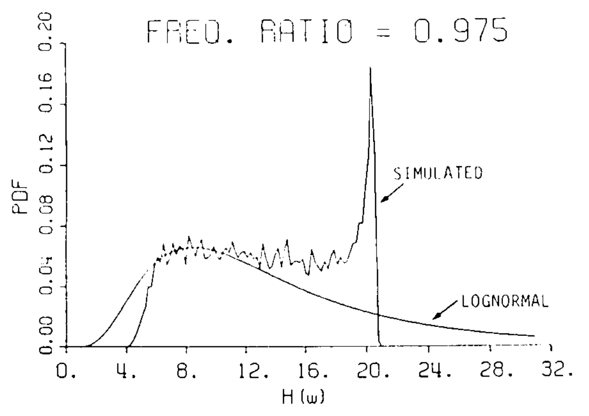
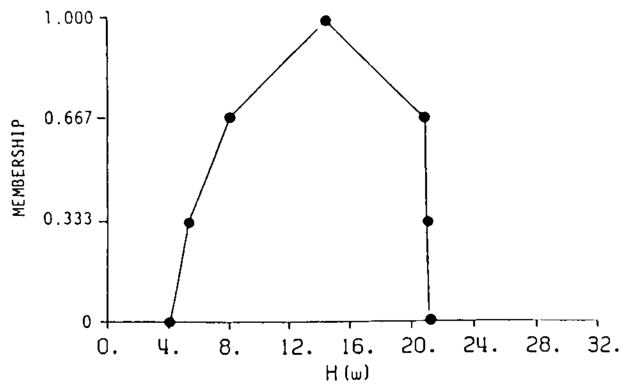


Figure 3. Membership Functions for FRF Amplitude (from [11])



(a) Probability Density Functions.



(b) Membership Function.

Figure 4. Bounding Quality of Fuzzy Sets near Resonance

N91-20697

THE AI BUS ARCHITECTURE FOR DISTRIBUTED KNOWLEDGE-BASED SYSTEMS

Dr. Roger D. Schultz and Iain Stobie

Abacus Programming Corporation, 14545 Victory Blvd., Van Nuys, CA 91411

Abstract

The AI Bus architecture is layered, distributed object-oriented framework developed to support the requirements of advanced technology programs for an order of magnitude improvement in software costs. The consequent need for highly autonomous computer systems, adaptable to new technology advances over a long lifespan, led to the design of an open architecture and toolbox for building large-scale, robust, production-quality systems. The AI Bus accommodates a mix of knowledge-based and conventional components, running on heterogeneous, distributed real-world and testbed environments.

This paper describes the concepts and design of the AI Bus architecture and its current implementation status as a Unix C++ library of reusable objects. Each high-level semi-autonomous agent process consists of a number of knowledge sources together with inter-agent communication mechanisms based on shared blackboards and message-passing acquaintances. Standard interfaces and protocols are followed for combining and validating subsystems. Dynamic probes or demons provide an event-driven means for providing active objects with shared access to resources, and each other, while not violating their security.

This work was carried out for the ALS STRESS (Space Transportation Systems Expert Systems Study) ADP 2301 & 2302, and resulted in a prototype implementation of many of the designed objects. It is now being used as the fundamental framework of Abacus' cooperative systems research project, which is examining various problem-solving mechanisms in distributed AI.

1. Introduction

The AI Bus was first developed as an approach to integrating the Space Station software, and more recently has been applied to the Advanced Launch Systems project (ALS). Both applications share requirements of a long life-time with several upgrades and high degrees of autonomy. Since the major cost in large modern software systems is that of maintenance, a major goal of the AI Bus is to provide a toolbox of reusable "plug-compatible" software objects.

In this paper we review the architecture's requirements, design and current implementation using Unix and C++. Since a particular interest is in supporting high-level models of cooperation and problem-solving, we also describe our current experimentation in these areas using the AI Bus facilities..

2. Requirenents

The AI Bus architecture was developed to meet the following requirements:

- Support cooperating, distributed systems
- Support embedded and real-time applications
- Facilitate technology upgrades during long lifetime
- Permit mixed procedural, knowledge-based and off-the-shelf components
- Support cooperation of autonomous components
- Suport control system and sensor-based applications
- Support fault-tolerant approaches
- Facilitate verification and validation

These requirements drove the design, as summarized in the Figure 1.

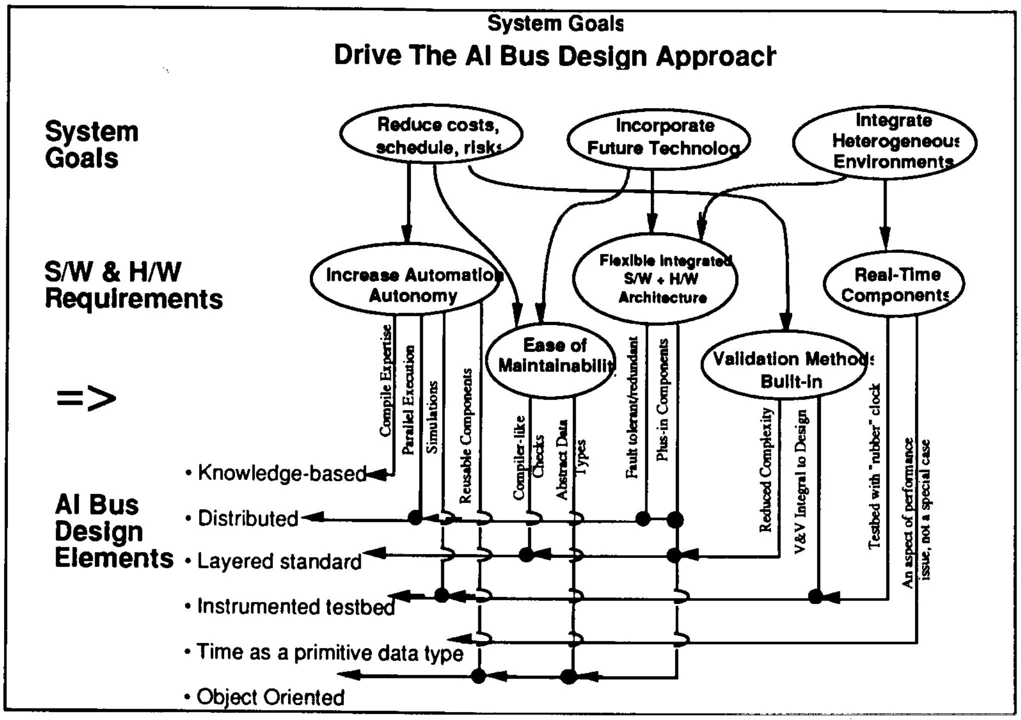


Figure 1—Origin of the AI Bus

3. Design

The design of the AI Bus is specified as an object library. The objects provide both an implied implementation architecture, and a standard interface specification. Unlike a jigsaw puzzle, where the pieces dictate one particular assembly architecture, the building block objects of the AI Bus must support multiple instances of architectures, being more like an erector set. A hardware analogy that comes close to the software framework provided by the AI Bus is that of a kit composed of a computer backplane, some standard function cards such as memory and processors, device interfaces, etc. and user customizable cards which contain the logic necessary to interface to the bus, and accept standard daughter cards, wire wrap, etc. to allow the application designer to construct an application specific function card. Application systems are then built by selecting the proper combination of the highest level cards that meet the application design requirements, and integrating them in a way to solve the application problem.

In addition to this object-oriented approach, a layered specification was developed: at the bottom are the physical entities, then the operating system components, then conventional tools such as databases and user interfaces, followed by knowledge-based tools such as inference engines, and at the top are generic applications such as diagnosis shells which simply need to be customized for a specific application. Services in one layer are insulated from changes in the implementation of services they use in lower layers, because the (public) protocols remain the same despite changes in the (private) implementation. Thus, alternative implementations can be selected and software upgrades can be installed without alteration of higher-level modules. Along with the protocols, a set of static verification tools checks the application syntax for possible errors. The separation of the representation language from its implementation permits modular V & V, as does the layered approach. Furthermore, dynamic validation is supported by the AI Bus audit probes, which are intelligent demons attached to the service objects, to be used in building an instrumented testbed. These audit probes act like stream transducers which monitor and query not only physical transactions (over the network or a database, for example) but also software invocations.

The layers and representative object classes are illustrated in Figure 2.

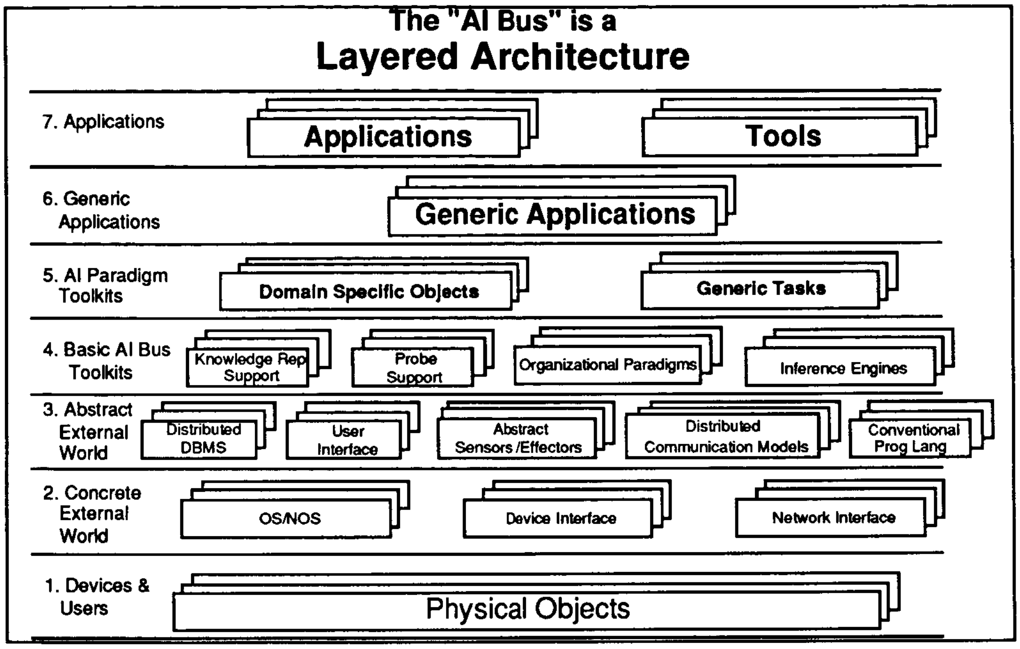


Figure 2. The layers of the AI Bus

4. Implementation

The design of the AI Bus was summarized in a set of abstract-data-type class specifications, intentionally kept language-independent in order to avoid restricting the design. As well as defining the interfaces, the design also specified inheritance between classes. For the implementation, we chose C++ and Unix because of the performance benefits of a relatively low-level language (as opposed to Smalltalk, for example) and its wide availability: a fundamental goal was to build a production quality system, not an experimental testbed. For the common knowledge representation language (layer 4) we chose Clips because it is distributed with source code and hence is amenable to customization. Message passing between distributed Clips systems was easily accomplished by writing three user-defined C++ functions (aibus_ask, aibus_tell, aibus_answer) that are called from the right hand side of a Clips rule. The communication services were built on top of the RPC protocol, and the user interface used X Windows.

We are currently working on decomposing the inference engine into object-oriented modules, so that rules can inherit conditions and actions, and rule-bases can inherit rules from other rule bases, and also incorporating non-linear fact and pattern representations (e.g. Prolog's recursive structures). At the lower layers, we are interested in non-Unix platforms and using commercial distributed operating systems.

5. Support for Cooperative Systems

Although developments in the last fifteen years have taken advantage of hardware advances by distributing data and processing, the control has remained centralized in master-slave relationships. Machines are now "talking" to one another, but the question for cooperative systems is deciding what to say, when, and by whose authority. Just as humans form organizations in order to function more effectively - the whole is greater than the sum of the parts - the promise of cooperative systems is that they can tackle problems beyond the capabilities of current architectures.

The AI Bus follows the distributed object oriented model of interaction between loosely-coupled agents, the fundamental active entities which communicate via messages (Ref. [1,2,3]). An agent is defined as a collection of knowledge sources and an organization; these knowledge sources may be implemented as expert systems (an inference engine and a knowledge base) or a conventional system - just so long as the specified interface is followed. One organizational mechanism is the blackboard, based on the paradigm of agents sharing their problem solving state (Ref. [4]). Each knowledge source has a list of capabilities and interests - which match questions it can answer and information it would like to be told - the agent advertises these attributes with the Finder (a lower layer communication object) and keeps a cache of other agents' capabilities and interests for subsequent communication.

An agent's specification thus permits implementation along several sizes of granularity. Internally, it can be a whole organization of problem solvers, or just a simple procedural program. For efficiency reasons in Unix-like environments where context-switching is costly, a large grain may be preferred, and this can be used at the next layer up as a generic task - an agent which is a specialist in one area of problem solving (Ref. [5]). An example of the internal organization of a complex agent is illustrated in Figure 3.

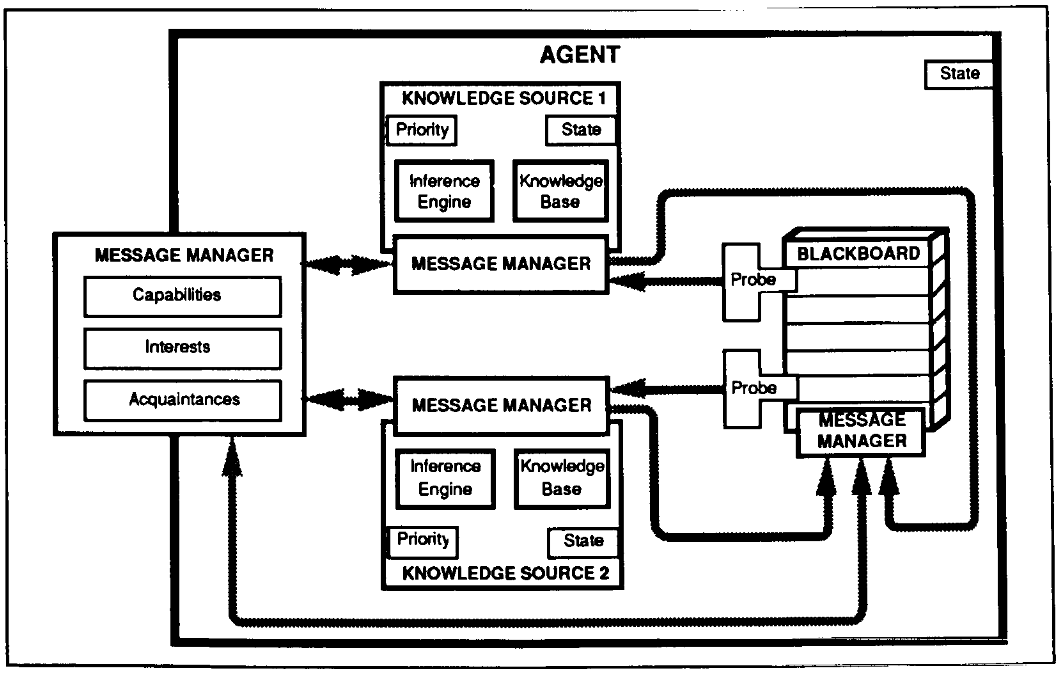


Figure 3. An Example of an Agent Composed of Several Layer 4 Objects

The AI Bus extends the paradigm of event-driven programming in its Probe object (Ref. [6,7]). A probe is activated based on matching patterns of events and conditions and routes information about subsystem activity to interested parties. A probe's history can be used to maintain partial matches for efficiency (e.g. in the blackboard), and has a priority for use in scheduling. A standard event, condition and action language allows the evaluation and interpretation of probes to be implemented by the probed object - a class of probeable objects is specified, and includes databases, network communication, blackboards and agents; there are corresponding subclasses of probes.

Probes can be used to support dynamic validation and to monitor resource usage. A subclass of probes called abstract sensor/effectors can be used in hierarchical process control applications - they provide data, retain state and do some filtering, but in addition they recognize alarm situations and provide direct pathways between each other for fast response.

A blackboard is realized in the AI Bus as a restricted subclass of agent - it is a passive server which is interested in everything (or at least whatever it is programmed for). Agents post information on the blackboard by sending it messages, they install probes on it to gather information resulting from matching events plus several current and historical conditions. A blackboard is thus a semi-permanent communication space, but also acts as a mechanism for loosely-coupled organization whereby several agents can combine partial results without repeated inter-agent communication. It is more than a global database, in that the probes' histories provide a short-term memory and record of partial matches, so that new additions and requests can be processed quickly (in the style of the Rete algorithm for rule-based systems); in contrast, database queries are processed one at a time. This is an object-oriented version of the blackboard concept, and it is important to contrast it with blackboard systems which contain a centralized scheduler in control of the serial execution of agents: in the AI Bus the agents are autonomous. Although logically centralized, a blackboard may be physically distributed for performance reasons: in this case, consistency must be maintained using techniques (e.g. multiple copies, deadlock avoidance) borrowed from distributed databases.

A layer of services exists between the operating system and the programming tools which allows the developers to concentrate on problem-solving rather than worrying about actual physical locations. Each agent has a Post Office object, which queues incoming messages and permits addressing by name, rather than location. The Post Office uses a distributed Finder to map globally unique names to the addresses of active objects. Control is passed via messages which represent remote procedure calls - they are intercepted by an agent's Message Manager, which is responsible for converting messages to procedures, and keeps a queue of questions received together with their askers (for subsequent direction of replies). Remote procedure calls can be asynchronous or synchronous. The question of whether the receiving agent blocks until it processes the request depends on the organization used: if the agent does, it is under the control of the sender (a client-server relationship), if not it is autonomous.

6. Current Direction

Having implemented a subset of the AI Bus objects, we are currently experimenting with using these tools in developing cooperative systems for applications such as air traffic control. In this domain, the decentralized controllers are assumed to be non-hostile, but nevertheless overall coherence and efficiency can deteriorate because of ill-informed local decisions. Facilities such as blackboards and negotiation protocols enable the controllers to make decisions based on more global understanding of the situations, and to cooperate on long-term solutions. Feedback on the usefulness of the tools developed so far has proved to be an esssential driver in the further development of the AI Bus framework.

References

1. Agha, Gul, *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1986.
2. Ishikawa, Yutaka and Mario Tokoro, *Orient84/K: An Object Oriented Concurrent Programming Language for Knowledge Represetnation* in Object Oriented Concurrent Programming, Yonezawa & Tokoro, eds, MIT Press, 1987.
3. Yonezawa, A. , J-P. Briot, E. Shibayama, *Object-Oriented Concurrent Programming in ABCL/1*, in Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.
4. Nii, Penny, *Blackboard Systems*, AI Magazine Volume 7, nos. 3 and 4, 1986.
5. Chandrasekaran, B., *Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design*, IEEE Expert, Fall 1986.
6. Schultz, Roger and A. Cardenas, *An Approach and Mechanism for Auidtable and Testable Advanced Transaction Processing Systems*, IEEE Transactions on Software Engineering, SE-13 (6), June 1987.
7. Schultz, Roger and A. Cardenas, *An Expert System Shell for Dynamic Auditing in a Distributed Environment*, ACM SIGSAC '87 Conference Proceedings, 1987.

ADVANCED AUTOMATION AND SOFTWARE

N91-20698

ART-Ada: An Ada-Based Expert System Tool

S. Daniel Lee and Bradley P. Allen

Inference Corporation
550 N. Continental Blvd.
El Segundo, CA 90245

Abstract

The Department of Defense mandate to standardize on Ada as the language for software systems development has resulted in increased interest in making expert systems technology readily available in Ada environments. NASA's Space Station Freedom is an example of the large Ada software development projects that will require expert systems in the 1990's. Another large-scale application that can benefit from Ada-based expert system tool technology is the Pilot's Associate (PA) expert system project for military combat aircraft. This paper describes ART-Ada, an Ada-based expert system tool. ART-Ada allows applications of a C-based expert system tool called ART-IM to be deployed in various Ada environments. ART-Ada is being used to implement several prototype expert systems for NASA's Space Station Freedom Program and the U.S. Air Force.

1. Introduction

The Department of Defense mandate to standardize on Ada as the language for software systems development has resulted in increased interest from developers of large-scale Ada systems in making expert systems technology readily available in Ada environments. Two examples of Ada applications that can benefit from the use of expert systems are monitoring and control systems and decision support systems. Monitoring and control systems demand real-time performance, small execution images, tight integration with other applications, and predictable demands on processor resources; decision support systems have somewhat less stringent requirements.

An example project that exhibits the need for both of these types of systems is NASA's Space Station Freedom. Monitoring and control systems that will perform fault detection, isolation and reconfiguration for various on-board systems are expected to be developed and deployed on the station either in its initial operating configuration or as the station evolves; decision support systems that will provide assistance in activities such as crew-time scheduling and failure mode analysis are also under consideration. These systems will be expected to run reliably on a standard data processor, currently envisioned to be an 80386-based workstation. The Station is typical of the large Ada software development projects that will require expert systems in the 1990's.

Another large-scale application that can benefit from Ada-based expert system tool technology is the Pilot's Associate (PA) expert system project for military combat aircraft [3]. Funded by the Defense Advanced Research Projects Agency (DARPA) as part of its Strategic Computing Program, the PA project attempts to automate the cockpit of military combat aircraft using Artificial Intelligence (AI) techniques. A Lisp-based expert system tool, ART (Automated Reasoning Tool), was used to implement one of the two prototypes built during Phase I. An Ada-based expert system tool can provide a migration path to deploy the prototype on an on-board computer because Ada cross-compilers are readily available to run Ada programs on most embedded processors used for avionics.

Inference has been involved with Ada-based expert systems research since 1986. Initial work centered around a specification for an Ada-based expert system tool [4]. In 1988, the ART-Ada Design Project was initiated to design and implement an Ada-based expert system tool [6], [10], [11]. At the end of 1989, ART-Ada was released to beta sites as ART-Ada 2.0 Beta on the VAX/VMS and Sun/Unix platforms [7]. In 1990, eight beta sites, four NASA sites and four Air Force sites, will be evaluating ART-Ada 2.0 for eight months by developing expert systems and deploying them in Ada environments. The objectives of the ART-Ada Design Project were two fold:

1. to determine the feasibility of providing a hybrid expert system tool such as ART in Ada, and

2. to develop a strategy for Ada integration and deployment of such a tool.

Both of these objectives were met successfully when ART-Ada 2.0 beta was released to the beta sites.

Inference Corporation developed an expert system tool called ART (Automated Reasoning Tool) that has been commercially available for several years [5]. ART is written in Common Lisp and it supports various reasoning facilities such as rules, objects, truth maintenance, hypothetical reasoning and object-oriented programming. In 1988, Inference introduced another expert system tool called ART-IM (Automated Reasoning Tool for Information Management), which is also commercially available [8]. ART-IM is written in C and it supports a major subset of ART's reasoning facilities including rules, objects, truth maintenance and object-oriented programming. ART-IM supports deployment of applications in C using a C deployment compiler that converts an application into C data structure definitions in the form of either C source code or object code. ART-IM's interactive development environment includes a graphical user interface that allows browsing and debugging of the knowledge base and an integrated editor that offers incremental compilation. ART-IM is available for MVS, VMS, Unix, MS-DOS, and OS/2 environments.

Our approach in designing an Ada-based expert system tool was to use the architecture of proven expert system tools: ART and ART-IM. Both ART and ART-IM have been successfully used to develop many applications which are in daily use today [1], [12], [13]. ART-IM was selected as a baseline system because C is much closer to Ada. While ART-IM's inference engine was reimplemented in Ada, ART-IM's front-end (its parser/analyzer and graphical user interface) was reused as the ART-Ada development environment. The ART-IM kernel was enhanced to generate Ada source code that would be used to initialize Ada data structures equivalent to ART-IM's internal C data structures, and also to interface with user-written Ada code. This approach allows the user to take full advantage of the interactive development environment developed originally for ART-IM. Once the development is complete, the application is automatically converted to Ada source code. It is, then, compiled and linked with the Ada runtime kernel, which is an Ada-based inference engine.

2. Overall Architecture

ART-Ada is designed to be used by knowledge engineers who may not be familiar with Ada. With minimum knowledge about Ada, they can still develop a knowledge base in a high-level language whose syntax most resembles that of Common Lisp. When the knowledge base is completed, Ada source code can be generated automatically by simply "pressing a button".

When this automatically generated Ada code is compiled and linked with the Ada library of the ART-Ada runtime kernel, an Ada executable image is produced. ART-Ada also provides extensive capabilities for Ada integration so that the knowledge base can be embedded in an Ada environment. It would be best if the knowledge engineer developing the knowledge base works with an Ada programmer who serves as a system integrator. ART-Ada would be most useful for those who must deploy in Ada environments (because of the Ada mandate) expert system applications already developed using tools that do not support Ada deployment.

The overall architecture of ART-Ada is depicted in figure 2-1. The knowledge base is developed and debugged using an interactive user interface that supports three main features; a command loop similar to the Lisp eval loop, a graphical user interface for knowledge base browsing and debugging, and an integrated editor for incremental compilation of the knowledge base. Any user-written Ada code can be integrated into the knowledge base by either calling it from a rule or invoking it as a method for object-oriented programming.

Once the knowledge base is fully debugged, it can be automatically converted into an Ada package for deployment. The ART-Ada runtime kernel is an Ada library, which is in essence an Ada-based inference engine. An Ada executable image is produced when the machine-generated Ada code and any user-written Ada code, if any, are compiled and linked with the Ada library.

3. Knowledge Representation

ART-Ada's key feature is the integration of rule-based representation and object-based (frame-based) representation. It supports three different programming methodologies:

- Rule-based Programming -- Rules opportunistically react to changes in the surrounding database. Rules can fire (execute) in an order

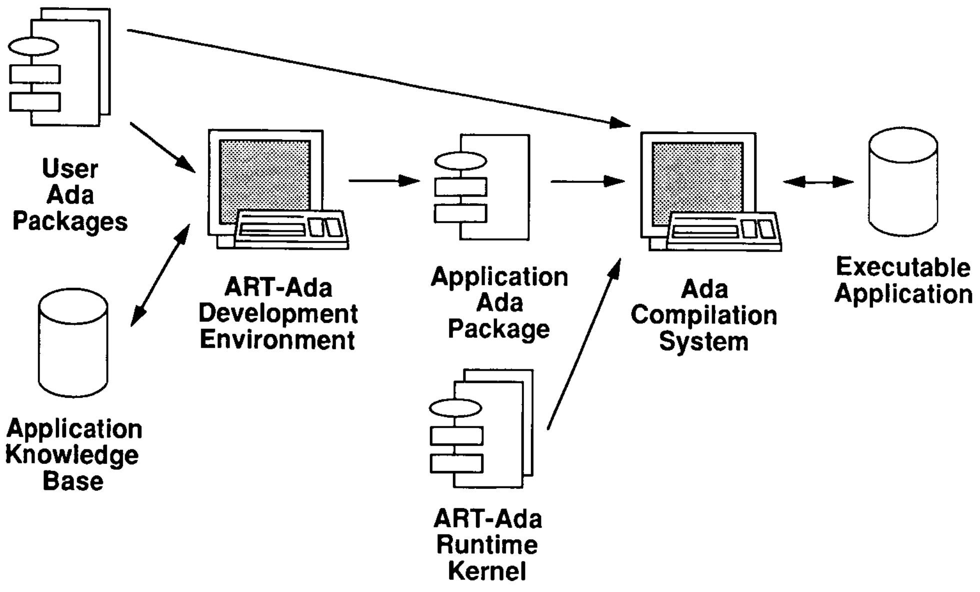


Figure 2-1: Overall Architecture of ART-Ada

based largely on the dynamic ordering of those changes. Rules cannot call other rules, and hence must communicate indirectly by making changes to the database which will, in turn, stimulate other rules.

- Object-Oriented Programming -- The fundamental unit of ART-Ada's object-oriented programming is the *object*, represented by a *schema*. Control is managed by sending *messages* to *objects* (schemas). The object reacts to the message by searching within itself for a *method* appropriate to that message. If an object does not have a method for the received message, it searches to see if it has inherited any appropriate methods from its parents. Once a method has been found, the object carries out the actions associated with the method.

- Procedural Programming -- ART-Ada's procedural language supports function calling, iteration (for, while) and conditionals (if, and, not). There are more than two hundred functions available in the procedural language.

ART-Ada's rule system is based on the optimized Rete pattern-matching algorithm [2]. Unlike OPS5, ART-Ada rules can pattern-match on objects called *schemas* as well as on lists called *facts*. *Facts* are similar to Lisp lists and do not support any inheritance. *Schemas* are similar to CLOS (Common Lisp Object System) objects; they are organized as attribute-value pairs and support *inheritance* through the *is-a* (subclass) and *instance-of* (member) relations. In the following example, *mammal* and *dog* are schemas while (animal-found dog) is a fact. *Mammal* is a class and *dog* is a subclass of the class *mammal*; they are linked with an *is-a* link. On the other hand, *fido* is a member of classes *dog* and *mammal*; it is linked to the class *dog* through an *instance-of* link. The significance of the relations *is-a* and *instance-of* is that the attribute-value pairs gets inherited either from a class to a subclass or from a class to a member. In the following example, *fido* will inherit attributes (eats meat), (socialization pack), (locomotion-mechanism run), and (instance-of mammal) from *dog*; it will also inherit (feeds-offspring milk) and (skin-covering hair) from *mammals.* As shown in the rule *determine-if-dog* that matches on both a schema pattern (schema ?animal (...)) and a fact pattern (classify-animal ?animal), the ART-Ada rules can match with schemas as well as facts. In order to optimize performance, ART-Ada uses two separate pattern matchers: one for schemas and one for facts.

```
(defschema mammal
   (feeds-offspring milk)
   (skin-covering hair))

(defschema dog
   (is-a mammal)
   (eats meat)
   (socialization pack)
   (locomotion-mechanism run))

(defschema fido
   (instance-of dog)
   (owned-by John))
```

```
(defrule determine-if-dog
   "Determine if subject is a dog."
   (classify-animal ?animal)
   (schema ?animal
      (is-a mammal)
      (socialization pack)
      (eats meat))
   =>
   (assert (schema ?animal
              (is-a dog)))
   (assert (animal-found dog)))
```

When an expert system deduces a conclusion (e.g. to diagnose faults in an electric circuit), it is often required to answer a question like "why?". This capability is called *explanation*. In ART-Ada, an explanation capability can be implemented using the *justification system*. When enabled, the justification system can provide a listing of the rules and data objects which were responsible for creating a particular fact or schema. By embedding features of the justification system in an application, the expert system can trace the steps leading to a particular conclusion. The justification system is also a powerful debugging tool when used during the development of an expert system. Should an application exhibit unexpected behavior during development, the programmer can exploit the features of the justification system to discover the source of the problem.

In the following example, if (classify-animal my-kangaroo) matches with a LHS pattern (classify-animal ?animal) where ?animal is a variable, and the rule fires to assert (schema my-kangaroo (is-a marsupial)), then we say that (classify-animal my-kangaroo) *justifies* (schema my-kangaroo (is-a marsupial)). In ART-Ada, consistency of the knowledge base is maintained by a justification-based truth maintenance system (JTMS) called Logical Dependencies. If *logical* is wrapped around (classify-animal ?animal), (schema my-kangaroo (is-a marsupial)) is not only *justified by* but also *logically dependent* on (classify-animal my-kangaroo); when (classify-animal my-kangaroo) is retracted from the knowledge base, (schema my-kangaroo (is-a marsupial)) is also retracted, and therefore consistency of the knowledge base is maintained automatically.

```
(defrule determine-if-marsupial
   "Determine if subject is marsupial."
   (logical (classify-animal ?animal))
   (schema ?animal
      (is-a mammal)
      (carries-offspring pouch))
   =>
   (assert (schema ?animal
              (is-a marsupial))))
```

In ART-Ada, object-oriented programming can be used with rule-based programming to take advantage of both paradigms. In the following example, the rule *print-out-object* is used to sent the *print* message to all objects that are instances of *object*. When an object *my-triangle* matches with the rule *print-out-object*, an inherited method *print-triangle* will be invoked. Methods can be defined either in ART-Ada's procedural language using *def-art-fun* which is similar to the Lisp defun, or directly in Ada using *def-user-fun* which will be discussed later.

```
;;; define objects

(defschema object
  (print print-unknown))

(defschema circle
  (is-a object)
  (print print-circle))

(defschema triangle
  (is-a object)
  (print print-triangle))

(defschema my-triangle
  (instance-of triangle)
  (position (1 2)))

;;; define a rule that sends a print message.

(defrule print-out-object
  (schema ?object
          (instance-of object)
          (position (?x ?y)))
  =>
  (send print ?object ?x ?y))
```

4. Knowledge Base Debugging

ART-Ada offers three main features in the user interface called the *Studio*:[*]

- a command loop,
- a graphical user interface, and
- an integrated editor.

ART-Ada's command loop is similar to the Lisp eval loop, in which user input is interpreted. More than two hundred functions are available in the command loop. Even Ada functions can be added to the command loop and called from the command loop.

[*]The Sun version supports only a command loop interface while the VAX/VMS version supports all three.

The Studio's interactive, menu-based graphical user interface provides immediate access to the knowledge base, and lets you monitor any aspect of program development or execution via an integrated network of menus and windows.

The Studio also provides a tightly integrated interface to the GNU Emacs full-screen editor. This interface facilitates the ART-Ada program development process by providing a number of powerful capabilities, such as incremental compilation of ART-Ada code.

The ART-Ada Studio can be used to do the following:

- Develop and execute an ART-Ada application.
- Browse the knowledge base -- to examine declarative (facts/schemas) knowledge, procedural (rules) knowledge, and runtime state, such as matches and activations.
- Debug the knowledge base -- by setting breakpoints in the programs and tracing their execution.
- Develop applications incrementally -- by editing the knowledge base to change facts or rules, or to modify program interactively.
- Generate Ada source code.

The ART-Ada/VMS Studio is based on DECwindows. The Studio is also implemented using other user interface standards (e.g. PM, OSF/Motif, ISPF) on other platforms.

5. Ada Integration

A major feature of ART-Ada is its ability to integrate expert systems technology with Ada. ART-Ada supports three types of Ada integration:

- *Ada call-out* refers to an ability to call Ada subprograms (procedures and functions) from the knowledge base (rules and methods).
- *Ada call-in* refers to an ability to call ART-Ada's public functions from Ada.
- *Ada call-back* is a special case of Ada call-in and refers to an ability to call ART-Ada's public functions from an Ada subprogram called from the knowledge base using Ada call-out.

Designers of expert systems will want to develop their own Ada code to provide user and system interfaces for their applications. There also may be a need to interface expert systems with other Ada applications (e.g. a signal processing application). A primary benefit of incorporating Ada code into the knowledge base is that Ada code will execute faster than similar code written in the ART-Ada procedural language. A consistent Ada call-in and call-out interface is provided for both development and deployment environments so that user-written Ada code runs without modification when it is deployed in Ada. In order to illustrate how an Ada subprogram is called from the knowledge base, let's consider the following rule:

```
(defrule distance-calculation-rule
  "calc distance between airfield and base"
  (schema ?airfield
    (instance-of airfield)
    (lat ?lat1)
    (lon ?lon1))
  (schema ?base
    (instance-of base)
    (lat ?lat2)
    (lon ?lon2))
  =>
  (bind ?distance
    ;; call an Ada function to calc distance
    (calculate-distance ?lat1 ?lon1
                        ?lat2 ?lon2))
  (assert
    (distance ?base ?airfield ?distance)))
```

The function, *calculate-distance*, can be implemented either in the ART-Ada procedural language or in Ada, but the Ada version would run faster. The ART-Ada construct *def-user-fun* specifies the interface between ART-Ada and Ada. It establishes an ART-Ada function name which calls out to the corresponding Ada subprogram, and it provides a description of data being passed. For example, *calculate-distance* can be specified as an Ada function as follows:

```
(def-user-fun calculate-distance
  :args ((lat1 :float)
         (lon1 :float)
         (lat2 :float)
         (lon2 :float))
  :returns :float
  :compiler :dec-ada)
```

This *def-user-fun* statement specifies that the ART-Ada function *calculate-distance* will call out to an Ada function CALCULATE_DISTANCE. There are four arguments of a type floating-point number being passed to Ada. The return value is also a floating-point number. It also specifies the default Ada compiler for the platform (i.e. DEC Ada). The corresponding Ada code should be declared in a package called *USER* and would look like:

```
-- ART is a public package of ART-Ada.
with ART;
-- USER is a package for user's Ada code.
package USER is

  function CALCULATE_DISTANCE
    (LAT1, LON1, LAT2, LON2 : ART.FLOAT_TYPE)
       return ART.FLOAT_TYPE;

end USER;
```

ART-Ada	Ada	Size
integer	INTEGER_TYPE	32 Bits
float	FLOAT_TYPE	64 Bits
boolean	BOOLEAN_TYPE	
string	STRING	
symbol	STRING	
art-object	ART_OBJECT	

Table 5-1: Data Types for Ada Call-in/Call-out

Ada data types supported for the call-in and call-out interfaces are: 32 bit integer (INTEGER_TYPE), 64 bit float (FLOAT_TYPE), boolean (BOOLEAN_TYPE), string and symbol (STRING), and an abstract data type for objects in ART-Ada (ART_OBJECT). Table 5-1 summarizes the mapping between ART-Ada and Ada data types.

6. Ada Code Generation

ART-Ada takes one or more ART-Ada source files as input and outputs Ada source files that represent a single Ada package. At any point after ART-Ada source files are loaded into ART-Ada and the knowledge base is initialized for execution, the Ada code generator may be invoked to generate Ada source code. An Ada package specification generated by ART-Ada for an example application called MY_EXPERT_SYSTEM is shown below:

```
-- generated automatically by ART-Ada
package MY_EXPERT_SYSTEM is

  -- initialize the application.
  procedure INIT;

end MY_EXPERT_SYSTEM;
```

A simple Ada main program that initializes and runs the application MY_EXPERT_SYSTEM is shown below. It is the simplest way to run an ART-Ada application in an Ada environment. It is possible, however, to embed it in a large Ada program. ART-Ada's public Ada packages, ART and SCHEMA, include a full set of Ada utilities to control and access procedurally the knowledge base from Ada. In OPS5, for example, it is hard to access working memory elements procedurally. In ART-Ada, Ada utilities are provided to access the knowledge base directly from Ada.

```
-- This is a main program written by the user.
-- ART is a public package of ART-Ada.
with ART, MY_EXPERT_SYSTEM;
procedure MAIN is
  TOTAL_RULES : ART.INTEGER_TYPE;
begin
  MY_EXPERT_SYSTEM.INIT;              -- initialize
  TOTAL_RULES := ART.A_RUN(-1);       -- run it.
end MAIN;
```

In addition to generating the Ada source code that initializes the knowledge base, a call-out interface module is generated as a separate procedure; it is a large case statement that contains all Ada subprograms called out to from ART-Ada. ART-Ada also generates a command file used to compile all Ada files generated by ART-Ada.

7. Ada Runtime Deployment

The steps needed to deploy an ART-Ada application in Ada are summarized below:

1. Develop and debug an application using ART-Ada's interactive development environment. If necessary, call out to Ada using the call-in/call-out interface.

2. Generate Ada code from ART-Ada using the Ada code generator. If the Ada compiler platform is different from the ART-Ada development platform, the generated Ada code can be moved to the platform on which the Ada compiler runs as long as the ART-Ada runtime kernel is available for that platform.

3. Compile the generated Ada code and user-written Ada code using either a self-targeted compiler or a cross-compiler into an appropriate Ada library of the ART-Ada runtime kernel.

4. Create an Ada executable image by linking an Ada main program.

5. Deploy the Ada executable image on a host computer or on a target system.

8. Future Work

According to a recent benchmark, ART-Ada does not perform as well as ART-IM. While immature Ada compilers also contribute to the poor performance, fundamental problems of the Ada language itself have been uncovered [9]. Some examples are:

- dynamic memory management,
- function pointers, and
- bit operators.

Among these, the overhead of dynamic memory management is the most serious problem. Due to the dynamic nature of expert systems, it is necessary to allocate memory dynamically at runtime in ART-Ada and ART-IM. The direct use of *new* and *unchecked_deallocation* is the only dynamic memory management method available in Ada. The problem with this method is that *new* incurs a fixed overhead associated with each call and it is called very frequently to allocate a relatively small block for an individual data structure. It results in a performance penalty in size and the slower execution speed. This is also aggravated by the poor implementation of *new* in the Ada compiler.

The existing Ada features, *new*, *unchecked_deallocation*, and *unchecked_conversion*, are too restrictive and totally inadequate for a complex system that requires efficient memory management. More flexible features (perhaps in addition to the existing ones) should be provided. This is particularly important in embedded system environments that impose a severe restriction on the memory size.

This issue and others were presented to several members of the Ada 9X Project in a meeting held in Washington, D.C. in March, 1990. We believe that they should be addressed by the Ada 9X standard. Unfortunately, the revised Ada language based on the Ada 9X will not be available until 1993 or later, which would be too late for the Space Station Freedom software development schedule.

Our current research effort is focused on improving the performance of ART-Ada by implementing ART-Ada's own memory manager using current technology. If it is not possible to implement it in Ada, we will implement it in another language (e.g. an assembly language). ART-Ada has an Ada code generator, which generates Ada code that relies on new and unchecked_deallocation. The current code generator would have to be redesigned to be compatible with the new memory manager.

Other Ada language issues such as function pointers, bit operators and portability and compiler problems encountered during the development of ART-Ada are discussed elsewhere [11], [9].

9. Acknowledgments

The authors wish to acknowledge the guidance and support of Chris Culbert and Bob Savely of NASA Johnson Space Center, Greg Swietek of NASA Headquarters, and Captain Mark Gersh of the U.S. Air Force. Mark Auburn, Don Pilipovich, Mike Stoler and Mark Wright of Inference Corporation contributed to the project.

References

1. Dzierzanowski, J.M. et. al. The Authorizer's Assistant: A Knowledge-based Credit Authorization System for American Express. Proceedings of the Conference on Innovative Applications of Artificial Intelligence, AAAI, 1989.

2. Forgy, C.L. "RETE: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem". *Artificial Intelligence 19* (1982).

3. Hugh, D.A. "The Future of Flying". *AI Expert 3*, 1 (January 1988).

4. Inference Corporation. Ada-ART, Specification for an Ada-based State-of-the-Art Expert System Construction Capability. Inference Corporation, August, 1987.

5. Inference Corporation. *ART Version 3.2 Reference Manual.* Inference Corporation, 1988.

6. Inference Corporation. ART/Ada Design Project - Phase I, Final Report. Inference Corporation, March, 1989.

7. Inference Corporation. *ART-Ada/VMS 2.0 Beta Reference Manual.* Inference Corporation, 1989.

8. Inference Corporation. *ART-IM/VMS 2.0 Beta Reference Manual.* Inference Corporation, 1989.

9. Lee, S.D. Toward the Efficient Implementation of Expert Systems in Ada. Submitted to the TRI-Ada Conference, ACM, 1990.

10. Lee, S.D., Allen, B.P. Deploying Expert Systems in Ada. Proceedings of the TRI-Ada Conference, ACM, 1989.

11. Lee, S.D., Allen, B.P. ART-Ada Design Project - Phase II, Final Report. Inference Corporation, February, 1990.

12. Nakashima, Y, Baba, T. OHCS: Hydraulic Circuit Design Assistant. Proceedings of the Conference on Innovative Applications of Artificial Intelligence, AAAI, 1989.

13. O'Brien, J. et. al. The Ford Motor Company Direct Labor Management System. Proceedings of the Conference on Innovative Applications of Artificial Intelligence, AAAI, 1989.

N91-20699

ADVANCED CLIPS CAPABILITIES

Mr. Gary Riley
Mr. Brian L. Donnell
Software Technology Branch
NASA Johnson Space Center
Mail Stop PT4
Houston, TX 77058

ABSTRACT

The 'C' Language Integrated Production System (CLIPS) is a forward chaining rule based language developed by NASA at the Johnson Space Center. CLIPS was designed specifically to provide high portability, low cost, and easy integration with external systems. The current release of CLIPS, version 4.3, is being used by over 2,500 users throughout the public and private community. The primary addition to the next release of CLIPS, version 5.0, will be the CLIPS Object-Oriented Language (COOL). The major capabilities of COOL are: class definitions with multiple inheritance and no restrictions on the number, types, or cardinality of slots; message passing which allows procedural code bundled with an object to be executed; and query functions which allow groups of instances to be examined and manipulated. In addition to COOL, numerous other enhancements have been added to CLIPS including: generic functions (which allow different pieces of procedural code to be executed depending upon the types or classes of the arguments), integer and double precision data type support, multiple conflict resolution strategies, global variables, logical dependencies, type checking on facts, full ANSI compiler support, and incremental reset for rules.

INTRODUCTION

The 'C' Language Integrated Production System (CLIPS) is a forward chaining rule-based production system developed by the Software Technology Branch at NASA/Johnson Space Center [1,2,3]. Version 4.3 of CLIPS has capabilities similar to those of OPS5 (Official Production System) and is syntactically similar to ART (Automated Reasoning Tool) [4,5,6]. Version 5.0 of CLIPS introduces several enhancements to version 4.3 which will be discussed in this paper.

CLIPS OBJECT-ORIENTED LANGUAGE

The primary addition to version 5.0 of CLIPS is the CLIPS Object-Oriented Language (COOL). COOL is a hybrid system incorporating various ideas from other Object-Oriented Programming (OOP) systems such as Smalltalk and the Common Lisp Object System (CLOS) [7,8,9,10]. Since other constructs within CLIPS (defrule, deffacts, etc.) were not originally developed in an object-oriented manner, no attempt was made to rewrite CLIPS to develop a completely object-oriented system. Thus, OOP features that have been added to CLIPS are extensions rather than fundamental changes to the entirety of CLIPS. For example, no attempt is made to let all CLIPS constructs be treated as objects, such as rules being instances of the rule class [10]. Instead, an imaginary dividing line was drawn with the class construct; once instances of a user-defined class are created, they may only be handled in an object-oriented manner, i.e. via messages. However, other elements of CLIPS, such as rules and facts, are still manipulated in the same (non-OOP) manner as they were in previous versions.

The primary features of COOL are: classes with multiple inheritance, instances, message-passing constructs [10], and a query system for determining and/or iterating an action over a set of instances which satisfy user-defined criteria. The message-passing constructs consist of around, before, primary, and after message-handlers as well as slot-accessor message-handlers and slot-daemons.

Objects

An object in CLIPS is defined to be one of the following: an integer or floating-point number, a symbol, a string, a multifield value, or an instance of a user-defined class. Objects may be used anywhere within CLIPS: expressions, facts, rule patterns, and so on. Objects are manipulated by sending them messages. Instances of a user-defined class can only be manipulated with messages, but other objects can be handled in a non-OOP manner as well. For example, two integers can still be added by calling the '+' function directly; sending a message to one of the integers with the other as an argument (as one would in Smalltalk) is unnecessary [9].

Instances are created with a special function called *make-instance*, which allocates the memory for an instance and then sends it the *init* message. All operations on objects which are instances of user-defined classes are done with messages. The message-passing concept used by COOL is similar to that of Smalltalk [9].

Classes

A class is a special construct in CLIPS, similar to rules. CLIPS does not support metaclasses (classes of classes) [7,8,9,10], since classes are not objects. Classes must be manipulated with special functions like other CLIPS constructs. For example, to print a rule, the function pprule is used, and, similarly, ppclass is used to print a class.

Classes in COOL are defined much in the same way as they are in CLOS. Full multiple inheritance is supported using the rules found in CLOS [7,8]. Classes can have any number of slots, and slots can have a list of facets selected from a predefined set. Some of the slot facets available are: single and multi-valued cardinality, static and dynamic default values, shared and local storage (similar to class and instance variables respectively in Smalltalk), and access restrictions.

Messages

Messages are implemented by pieces of procedural code written in CLIPS called message-handlers. These handlers are bundled with the class definitions, and thus inheritance relationships may be used to determine to which messages an instance can respond. The implementation of a message can be further subdivided into handler types: around, before, primary, and after. This notion is borrowed from generic function methods in CLOS [7,8]. Around handlers are meant to set up an environment within which the other handlers may execute. Before and after handlers perform auxiliary work outside the scope of the primary handler. The primary handler is intended to do the core of the work of the message. Within the body of a message-handler is the only place where the slots of an object can be directly accessed without using messages. However, an object may send other messages (including ones to itself) in the course of executing a message. The declarative flow of execution for COOL message-handlers is similar to the standard method combination type in CLOS [7,8]. COOL also provides imperative control by allowing handlers to explicitly call other handlers that they are shadowing.

Slot-daemons

For every slot in an instance, two implicit primary message-handlers are defined: one for reading the slot and one for the writing the slot. Users must use these messages to access explicitly the object's slots. Slot-daemons may easily be defined by defining around, before, or after message-handlers which correspond to these messages.

Instance-Set Queries and Distributed Actions

At present, only facts can be pattern-matched on the left-hand side of rules; pattern-matching against the state of an instance of a user-defined class is not possible. However, COOL does provide a useful query-system for determining and performing actions on sets of instances that meet certain user-defined criteria. This query-system can be used with control facts to accomplish a brute-force instance pattern-match. (Control facts and slot-daemons may also be used to this end.)

An instance-set is an ordered collection of instances. Each member of this set is an instance of a set of classes defined by the user. The set of classes can be different for each instance in the instance-set. For example, one instance-set definition might be the ordered pairs of men or boys and women or girls. If there is one instance of each of these four classes, then there would be four instance-sets which satisfy the definition: (Man-1,Woman-1), (Man-1,girl-1), (boy-1,Woman-1), and (boy-1,girl-1).

A query is a user-defined boolean expression applied to an instance-set to determine if the instance-set meets further user-defined restrictions. Continuing the example above, one query might be that the two instances in an ordered pair have the same age.

A distributed action is a user-defined expression evaluated for each instance-set which satisfies a query. Continuing the example above, one distributed action might be to simply print out the ordered pair to the screen.

Several different functions are provided in this system: determine if there are any instance-sets which satisfy the query, group and return all instance-sets which satisfy the query, perform an action for all instance-sets which satisfy the query, and others.

GENERIC FUNCTIONS

In addition to the object system itself, CLIPS 5.0 also supports generic functions [7,8,10]. Generic functions are groups of procedural code written in CLIPS that can later be called like any other CLIPS function. Different methods can be defined for generic functions that do different things depending on what the classes of the generic function arguments are. This allows new generic functions as well as standard CLIPS system functions to be overloaded. Although generic functions are not part of COOL (and can be used independently of it), they will utilize the full inheritance information of the classes of their arguments.

Generic functions in COOL are quite similar to generic functions in CLOS [7,8]. One difference is that COOL supports only primary methods, whereas CLOS has around, before, and after methods. This notion of splitting tasks into around, before, primary, and after parts was moved to messages and taken away from generic functions in COOL because it was felt intuitively that, for a particular set of arguments, a generic function should only execute one piece of code. However, it seemed reasonable that the implementation of a message might in fact be comprised of many different pieces of code.

CLIPS system functions which are not overloaded by generic functions completely bypass the generic dispatch mechanism. Thus, previous CLIPS programs will not pay any performance penalties simply as a result of the generic dispatch being available.

The argument restrictions which are used to determine the applicability of a method to a particular generic function call are somewhat more powerful than what is found in CLOS [7,8]. The user can specify that a restriction be any one of a list of classes, whereas CLOS only lets the user specify one class. Also, in COOL, the user may also specify an arbitrary boolean expression that the argument must satisfy for the method to be applicable. This is more powerful than CLOS individual methods, for they only allow the user to restrict the specific object rather than allowing any boolean expression. As a result of these enhancements, the precedence determination between methods in COOL is slightly more complicated than it is in CLOS.

To define new non-overloaded functions in CLIPS, the deffunction construct can be used in place of generic functions. Deffunctions allow a piece of procedural code to be written and used in the CLIPS language without any coding in an external language such as C. In previous versions of CLIPS, to add a new function, the user had to write it in C (or another language such as FORTRAN or Ada) and compile it, then relink CLIPS with the new function.

GLOBAL VARIABLES

The defglobal construct allows global variables to be defined which may then be accessed or set by rules, generic functions, and other constructs. Global variables allow information to be stored outside of facts (thus avoiding potentially unwanted pattern matching). For example, a global variable could be used to count the number of facts of a particular type. Incrementing the global variable from a counting rule would not reactivate the counting rule, whereas incrementing a value from a fact storing the count would retrigger the rule since the count fact would have to be matched against in the antecedent of the rule.

INTEGER DATA TYPE SUPPORT

CLIPS now supports an integer data type (represented internally as a C long integer). Floating-point numbers are now represented internally as C double precision numbers for greater accuracy. Previously, CLIPS stored all numbers as single precision floating-point numbers. Arithmetic functions such as addition, subtraction, multiplication, and division support mixed mode operations on integers and floats.

CONFLICT RESOLUTION STRATEGIES

Past versions of CLIPS supported a single conflict resolution strategy [4,11]. The order of rules to be executed on the agenda (the list of rules that have their conditions satisfied) was determined by the salience of the rule (a numerical value between -10,000 and 10,000) and the order of activation of the rule. Rules with higher salience are executed before rules with lower salience. Among rules of equal salience, the rule last activated is executed first (a "depth-first" or "stack" strategy).

CLIPS 5.0 now supports seven different conflict resolution strategies: depth, breadth, LEX, MEA, simplicity, complexity, and random. These resolution strategies are used to determine placement of an activation of a rule on the agenda between rules of equal salience. The depth strategy implements the "stack" placement (last-activated, first-executed) of activations found in previous versions of CLIPS. The breadth strategy implements a "queue" placement (first-activated, first-executed) of activations. The LEX and MEA strategies are similar to the OPS5 strategies of the same name [4,5]. The simplicity strategy executes activations of rules with simple antecedents before rules with complex antecedents. The complexity strategy works in a directly opposite manner to simplicity strategy. The complexity of the antecedent of a rule is determined by manner factors including the number of patterns, the number of constant comparisons, and the number of variable comparisons. The random strategy randomly determines the order of activations of equal salience. The conflict resolution strategy can be dynamically changed and the agenda will be updated to reflect the new strategy.

SALIENCE EXTENSIONS

The salience declaration within a rule is no longer limited to strictly integer constants. The declared salience for a rule can be an expression which references global variables as well as calling system and/or user defined functions. In addition, evaluation of salience values can now occur at several different times: when a rule is defined, when an activation is placed on the agenda, and every cycle of execution. The user also has the ability to refresh the salience values of activations on the agenda at any time.

DEFTEMPLATE FIELD CHECKING

The deftemplate construct introduce in version 4.3 of CLIPS provided a method for structuring facts by tagging each field of the fact with a name. This provided CLIPS with a record structure similar to procedural programming languages. Optional fields in the deftemplate construct allowed type, value, and range restrictions to be specified. However, only the CLIPS Cross Reference, Style and Verification (CRSV) utility tool was able to make use of this information. CLIPS 5.0 now supports type, range, and value checking for deftemplates both statically (when patterns or actions using deftemplates are loaded) and dynamically (when deftemplate facts are asserted).

LOGICAL DEPENDENCIES

Truth maintenance [4,12] is now supported in CLIPS through the use of logical dependencies. The "logical" pattern operator can be placed around the first N patterns of a rule to indicate that facts asserted by this rule are dependent upon the existence of the facts matching the logical patterns (or non-existence of facts matching negated logical patterns). A fact asserted from a rule with logical patterns is logically supported by that rule. A fact may have multiple logical support from the same or different rules. A fact asserted from a source other than a rule with logical patterns is unconditionally supported (and cannot be retracted as a result of truth maintenance). Whenever a fact is retracted that matched a logical pattern of a rule (or a fact is asserted that matched a negated logical pattern), the logical support from that rule for any fact asserted by that rule is removed. Any fact that loses all of its logical support is automatically retracted.

INCREMENTAL RESET

In previous versions of CLIPS, newly defined rules were not activated on currently existing facts. That is, rules were only activated based on facts that were added after the rule was defined. Thus it was not possible to load new rules into the system and have these rules activated on previously asserted facts without somehow reasserting those facts. Newly defined rules in CLIPS 5.0, however, are fully activated by currently existing facts. This makes it possible to dynamically load new rules and have them automatically updated based on the current set of facts.

The build function allows construct to be dynamically created during execution. This makes it possible for a rule to create new rules (which would be incrementally reset based on the currently existing facts). It is also possible to safely delete rules as one of the actions of the consequent of a rule. It is even allowed to delete the currently executing rule and have the actions of its consequent execute to completion.

ANSI COMPILER SUPPORT

In addition to the numerous features added to CLIPS 5.0, the source code has been modified to be ANSI C conformant wherever discrepancies occurred between "K&R C standards" and ANSI C standards. Function prototypes have also been used for all functions to increase the maintainability of the code. ANSI C features not compatible with K&R C compilers can be removed through the use of compiler directive flags.

CONCLUSION

Version 5.0 of CLIPS provides several new capabilities which significantly increase the usefulness of the tool. Among these capabilities are: object-oriented programming, generic functions, global variables, integer data type support, additional conflict resolution strategies, salience extensions, type, range and value checking for deftemplates, incremental reset, and logical dependencies.

REFERENCES

1. Culbert, C., *CLIPS Reference Manual.* NASA document JSC-22948, July 1989.

2. Riley, G., *CLIPS Architecture Manual.* NASA document JSC-23047, May 1989.

3. Giarratano, J., *CLIPS User's Guide.* NASA document, August 1989.

4. Brownston, L., Farrell, R., Kant, E. and Martin, N., *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming,* Addison-Wesley, 1985.

5. Forgy, C., *OPS5 User's Manual.* Department of Computer Science document CMU-CS-81-135, Carnegie-Mellon University, Pittsburgh, PA. 1981.

6. *ART Reference Manual,* Inference Corporation, Los Angeles, CA. 1986.

7. Keene, S., *Object-Oriented Programming in Common Lisp.* Symbolics, Inc., 1989

8. Bobrow, D., DeMichiel, L., Gabriel, R. , Keene, S., Kiczales, G. and Moon, D., *Common Lisp Object System Specification.* X3J13 document 88-002R, June 1988.

9. Pinson, L. and Wiener, R., An Introduction to Object-Oriented Programming and Smalltalk. Addison-Wesley, 1988.

10. Tello, E., Object-Oriented Programming for Artificial Intelligence. Addision-Wesley, 1989.

11. Cohen, P. and Feigenbaum, E. (ed.), The Handbook of Artificial Intelligence, Vol. I, William Kaufmann, Inc., 1982.

12. Cohen, P. and Feigenbaum, E. (ed.), The Handbook of Artificial Intelligence, Vol. II, William Kaufmann, Inc., 1982.

N91-20700

Parallel Processing and Expert Systems

Jerry C. Yan
Sterling Federal Systems Inc.,
NASA Ames Research Center,
MS 244-4, Moffett Field, CA 94035
(415) 604-4381
jerry@pluto.arc.nasa.gov

Sonie Lau
Information Sciences Division,
NASA Ames Research Center,
MS 244-4, Moffett Field, CA 94035
(415) 604-4944
lau@pluto.arc.nasa.gov

Abstract

Whether it be monitoring the thermal sub-system of Space Station Freedom, or controlling the navigation of the autonomous rover on Mars, NASA missions in the 90's cannot enjoy an increased level of autonomy without the efficient implementation of expert systems. Merely increasing the computational speed of uniprocessors may not be able to guarantee that real-time demands are met for large expert systems. Speed-up via parallel processing must be pursued alongside the optimization of sequential implementations. Prototypes of parallel expert systems have been built at universities and industrial laboratories in the US and Japan. This paper surveys the state-of-the-art research in progress related to parallel execution of expert systems. The survey is divided into three major sections: (i.) multiprocessors for parallel expert systems, (ii.) parallel languages for symbolic computations and (iii.) measurements of parallelism of expert systems. Results to date indicate that the parallelism achieved for these systems is small. The main reasons are: (i.) the body of knowledge applicable in any given situation and amount of computation executed by each rule firing are small; (ii.) dividing the problem solving process into relatively independent partitions is difficult; and (iii.) implementation decisions that enable expert systems to be incrementally refined hamper compile-time optimization. In order to obtain greater speed-ups, *data parallelism* and *application parallelism* must be exploited.

1. Introduction

The science and engineering objectives of NASA missions in the 90's cannot be met without an increased level of autonomy for both onboard and ground-based systems. For example, with *Mars Rover Sample Return*, the long delays associated with signal transmission between Mars and Earth require the Rover to make intelligent decisions and operate autonomously in real-time. The day-to-day operation of *Space Station Freedom* also depends critically on real-time expert systems — whether it be operating the thermal control sub-system, or flight tele-robotic servicers. Current implementations of expert systems run too slow. Merely increasing the computational speed of uniprocessors may not be able to guarantee that real-time demands be met for large systems. Speed-up via parallel processing must be pursued alongside the optimization of sequential implementations.

Parallel expert systems has been investigated at universities and industrial laboratories in the US and Japan. Prototypes of multiprocessors specifically designed for expert systems have been built. Results to date indicate that only certain applications are amenable to parallelization. In most cases, the degree of parallelism achieved is less than 10. In order to obtain higher speed-up values, we must understand why expert systems are difficult to parallelize, how they should be written and partitioned to obtain maximum parallelism, and how they can be effectively mapped onto parallel architectures.

In order to address these issues adequately, a survey of current state-of-the-art in parallel processing for expert systems has been carried out. Section 2 begins with a description of well known symbolic computation paradigms and state-of-the-art sequential implementation for them. Section 3 surveys four parallel hardware architectures specifically proposed for symbolic computation: DADO, NETL, the connection machine, and PIM. Section 4 surveys various parallel extensions to existing symbolic programming languages — parallel LISPs, CParaOPS5, concurrent PROLOG, and object-oriented languages. Section 5 reports the inherent parallelism observed in expert systems today and suggests why parallelizing expert systems is difficult. Finally, section 6 discusses how expert systems might be parallelized and what reasonable research directions might be.

2. Sequential Expert System Implementation

2.1 Software and Hardware Requirements

Unlike conventional software, expert systems operate on *symbols*, as well as *numbers*. Problem state information and problem solving knowledge are represented by data *structures* (or *shapes*) as well as *values*. As the problem solving process proceeds, arithmetic operations as well as *pointer manipulation* are performed by the hardware — creating new data structures, discarding old ones and changing the values, sizes and shapes of existent structures. Many paradigms have been proposed to represent problem solving knowledge and state information for this kind of computation. For example, knowledge may be represented *declaratively* (e.g. using *predicate calculus*) and processed based on *resolution*, simple *rules of inference*, *backward* and *forward* chaining. Knowledge may also be encoded *procedurally* (as programs) or *structurally* (as *semantic nets*). *Frames* and *objects* combine both representation techniques by attaching procedures to structured data.

Languages proposed for symbolic computations include list processing languages (e.g. Common Lisp), object-oriented languages (such as SMALL-TALK), and logic programming languages (e.g. Prolog). In order to implement these languages efficiently, new requirements are placed on compilers, operating systems and hardware architectures originally optimized to support arithmetic operations on data cells. Perhaps the most demanding language feature is the ability to construct, modify and access complex data structures dynamically during run-time. In order to support dynamic data structures, storage must also be allocated/reclaimed efficiently and transparently at run-time.

The Von-Neumann computer does not support this kind of (symbolic) computation directly. Hardware features supporting run-time type checking, garbage collection and pointer manipulation/arithmetic have been incorporated into Lisp and Prolog machines to facilitate the efficient implementation of expert systems.

2.2 Lisp Machines

Lisp and object-oriented languages have been efficiently implemented on Lisp machines (such as Symbolics 3600™, XEROX 1100™ and TI Explorer™). Hardware features designed specifically to enhance the performance of symbolic computations include: tagged memory architecture and processing hardware and hardware stacks. Lists are efficiently represented using *cdr-coding* schemes. Object-oriented programs also execute efficiently because slot value access, message processing, and class inheritance and mixing are implemented with very low overhead.

2.3 Prolog Machines

Sequential execution of logic programs such as Prolog have been greatly improved by the concept of the Warren Abstract Machine (WAM) [1]. Many of these ideas were studied and incorporated by the Japanese Fifth Generation Computer System (FGCS) project — the initial stage of which resulted in the development of the Personal Sequential Inference (PSI) machine rated at 30K LIPS (logical inferences per second). It incorporated UNIRED [2], a hardware accelerator, to increase the speed of *unification* and *reduction*.

3. Multiprocessors for Expert Systems

Given all these "state-of-the-art" enhancements mentioned in section 2, execution of large expert systems is still unable to meet the requirements of many applications such as air-traffic control, pilot's associate and real-time speech understanding. Multiprocessing must be pursued, together with innovations in software implementation, sequential hardware architecture and device technology, in order to speed-up expert systems. The next two sections summarizes the major developments in hardware and programming languages for parallel symbolic computing. Four machines are described in this section: DADO, NETL, the *connection machine*, and PIM.

3.1 DADO

The processing elements (PEs) of **DADO** [3] are connected as a binary tree. *Matches* and *updates* are processed in parallel based on simple broadcasts up and down the tree. Each PE has a special I/O device that performs three global operations (BROADCAST, REPORT, and MAX-RESOLVE) efficiently. A PE may execute instructions in its local memory and enlists its descendents by BROADCASTing to them. Each descendent executes instructions received and REPORTs back. The final solution may have to be determined by performing the MAX-RESOLVE function on the parent node's result and the two returned from its descendents.

Production systems were mapped onto DADO by dividing the binary tree into three logical layers. The top layer serves as a "decision maker"; it performs *synchronization*, *conflict-resolution* and the *act* phases. Productions are distributed across the next layer where the *match* phase and *instantiations* take place. The bottom layer holds the working memory elements. In order to reduce the communication bottleneck between peer nodes on different halves of the tree, data was duplicated wherever needed; this introduced consistency problems. Two prototypes were proposed [4] — the first of which, DADO1, consists of 15 PEs rated at 4 MIPS each. The speed up obtained on DADO1 was limited mainly because different tasks on different nodes require different processing times.

3.2 NETL

NETL [5] is a fine-grain SIMD machine. Its PEs can be logically interconnected as nodes in a semantic net. Parallel reasoning on NETL was performed via *marker passing* [6]. *Tokens* are sent through *nodes* (i.e. PEs) that lead to the solution. When a *token* goes through a node, a bit at the node is set. When the goal is reached, the node with the bit set constitutes the search space. For example, a node satisfying all the preconditions of a production could be located by propagating the preconditions concurrently through the network. The node with a bit set for each precondition would be the one that satisfies the rule.

3.3 Connection Machine

The PEs of the **Connection Machine** [7] are connected as a hypercube. All PEs execute in a lock-step manner based on an external clock and instructions from a front-end host computer. A set of flags on each PE can be selectively set — thereby giving more flexibility and expressiveness in the host computer's control. The performance of CM depends on the size and interdependencies of the data. Because the PEs have small local memories, data can be spread out over several PEs; thereby, requiring several communication steps to process a single piece of data.

3.4 Parallel Inference Machine

As specified by Japan's FGCS project overview [8], the overall target performance of the **Parallel Inference Machine (PIM)** is 10 to 20 million reductions per second (RPS). The pilot machine PIM/P, with 128 PEs connected as a hypercube, executes 50ns cycles in a four stage pipeline. Multiple PSIs have been networked together forming multiprocessors to test parallel system software eventually to be executed on PIM [9]. These include (i.) *Kernel Language Version 1* (KL1) — a parallel Prolog-like language based on *flat guarded horn clauses*, (ii.) a *multiple reference bit* scheme for local garbage collection, (iii.) a *weighted export count* to support inter-PE garbage collection., and (iv.) a *weighted throw count* scheme for terminating remote processes. Dynamic load balancing strategies on PIM are currently being researched.

4. Parallel Languages for Expert Systems

The parallel symbolic languages surveyed in this section (parallel LISPs, PROLOG, and *object-oriented* languages) augment existent languages with parallel constructs.

4.1 Parallel LISPs

QLISP [10] (queue-based multiprocessing Lisp) was designed to execute on shared-memory architectures. A scheduler assigns new processes on a global queue to the least busy processor in a *round-robin* fashion. The degree of multiprocessing can be controlled explicitly at run-time. Very few extensions are made to Lisp although some existent constructs take on new meanings in a multiprocessing setting. Processes are created using two constructs: QLET and QLAMBDA. QLET expresses parallelism that has regularity, for example, over an underlying data structure. QLAMBDA creates *closures* dynamically for expressing less regular parallel computations. QLISP runs currently on Encore multiprocessors.

MULTILISP [11] is an extension to *Scheme* with constructs supporting parallel execution. It provides lexical scoping as well as "first-class citizenship" for Lisp functions — which enables functions to be passed and returned as values (to other functions which may reside on other processors), or stored as part of a data-structure. The construct "(***future** body*)", creates a process to evaluate *body* and returns a *future* which acts as a *place holder* for (or a *promise* to deliver) the result of the evaluation. While the evaluation proceeds, the *future* can be used for constructing data structures or passed around as arguments. Any process which actually requires the value of the *future* will be suspended unless the evaluation process has completed. A "delay" construct is also provided to support *lazy-evaluation* — allowing a *future* to be evaluated only on demand. MultiLisp is implemented on the Butterfly and Concert [12].

A fine-gain version of parallel LISP called ***LISP** (previously known as CmLisp) is implemented on the Connection Machine. Operations can be performed simultaneously over each element of a large data structure. Some of the concurrent operations available are: *combine, create, modify,* and *reduce*. New SIMD-parallel operations can also be defined based on these concepts.

4.2 Parallel PROLOGs

Four sources of parallelism (and combinations of these) can be exploited in Prolog:

- *Or-parallelism*: Each rule, whose head unifies with a fact, can be solved in parallel.
- *And-parallelism*: Processes execute in parallel to solve each clause of the body.
- *Stream-parallelism* is a pipelined form of AND-parallelism. Unifications for the first sub-goal are forwarded to the process working on the next one as soon as it becomes available and so forth.

- *Search-parallelism*: Assertions are grouped so that search may proceed in parallel without contention to a single resource.

Two models which exploit some of these sources of parallelism have been proposed.

The **AND/OR** parallel execution model [13] provides a method for partitioning a logic program into small asynchronous and logically independent processes. A tree of processes is built as computation proceeds. *Start, redo,* and *cancel* messages are sent from parents to children who reply either with *success* or *fail* messages. In this model, an OR-process replaces the backtracking in sequential computation by acting as a *message center*[1]. It also filters out duplicate solutions by maintaining a list of successful messages from its children and messages sent to its parent. A parallel AND-process is more complicated because distributing literals across PEs has its problems[2] — the solutions to some of which were presented elsewhere [14].

The second model, **RAP-WAM** [15], was based on DeGroot's Restricted-And-Parallelism (RAP) work [16] and parallel extensions to WAM. RAP reduces the overhead associated with managing variable binding conflicts between goals. Previous approaches were unsatisfactory — compile-time approaches required user input on the variables while run-time approaches, such as the AND/OR model, were complex and expensive. RAP analyzed the clauses at compile-time and performed simple checks on the variables at run-time [17]. RAP-WAM also performed search with minimal backtracking by representing the problem as a condition graph to evaluate/analyze possible paths to select the best solution. This analysis also provided dependency information among goals.

4.3 Parallel *Object-Oriented* Languages

The performance of two *object-oriented* languages for distributed-memory architectures were studied by simulation by researchers at Stanford University. CAOS [18] computations consists of large grained asynchronous multiprocessing objects. Various message-sending primitives were defined, including synchronous and asynchronous SEND*s*, and SEND*s* which returned *futures*. LAMINA [19] provided extensions to LISP to support functional , object oriented, and shared variable styles of programming. Its implementation was based *stream* — a data type used to express pipelined operations by representing the promise of a (potentially infinite) sequence of values. These languages supported two concurrent problem solving frameworks developed based on the blackboard problem solving model. *Cage* and *Poligon*, were proposed for shared- and distributed-memory architectures respectively [20, 21].

5. Measuring Parallelism in Expert Systems

Parallel implementation of production systems (based on OPS5) have been extensively studied at Carnegie-Mellon University. Besides obtaining speed-up via parallel implementations of each phase, further speed-up may be obtained by allowing execution between phases to overlap (i.e. occur simultaneously). Nevertheless, because of the observation that 90% of processing time is spent in the *match* phase, their efforts focused on parallel implementations of the RETE-*match* algorithms [22]. Three parallel implementations were proposed [23]:

- *production parallelism* — rules fired concurrently;
- *node parallelism* — each node of the RETE-network fired concurrently;
- *intra-node parallelism* — the processing of each token to a two-input node of the RETE-network occurred concurrently;

These implementations, of decreasing granularities, subsumed one another and produced increasing levels of speed-up. Further speed-ups were obtained when changes to working memory are allowed to occur concurrently. Speed-up values of 6.3 to 12.4 were observed depending on the application.

5.1 Parallelism in Production Systems and Flat Concurrent Prolog Systems

Based on detailed measurements on six expert systems containing up to 1100 rules (written in OPS5) [23], three important observations were made:

A. Very few changes were made to working memory per recognize-act cycle. The number of RETE network nodes affected by changes to the working memory was small.

B. The total number of node activations per change was quite independent of the number of productions in the production-system program.

C. Variation in processing requirements for the (few) affected productions was large.

[1] It distributes work among its own children and sends the first successful tuple received back to its parent. Meanwhile, its other children continue working and *success* messages collected are only sent up to the parent if a *redo* message is received. Eager evaluation is implemented by sending *redo* messages to successful children so that more solutions are computed if the parent should require it. If no child succeeds, a *fail* message is returned to the parent.

[2] e.g. resolving binding conflicts among the literals, idle time waiting for literals to be bound; and some literals fail if attempts are made to solve them before certain variables are instantiated

These observations were explained as follows:

A. Firstly, an expert system contains a large body of knowledge about many different types of objects and diverse situations. The amount of knowledge (therefore, number of rules) associated with any specific situation is expected to be small. Secondly, most working-memory elements only describe a few aspects of a single object or situation; therefore, they could only be of interest to a few rules.

B. Programmers recursively divide problems into subproblems when writing large programs. The size of these subproblems are independent of the size of the original problem; it depends only on the complexity that the programmer can deal with at one time.

C. Rules accounting for different situations, formulated based on different heuristics, obviously exhibit different complexity and require different amount of processing.

These observations (and explanations) are not only specific to systems written in OPS5; they transcend all expert systems. For example, measurements on *flat concurrent prolog* systems also revealed that although the number of goals which exist at some point during execution may exceed 1000s, the average number of goals available for concurrent processing for most of the time is much smaller (< 12) [24]. These observations suggest major obstacles as far as obtaining speed-up for expert systems from parallel processing.

5.2 Obtaining Speed-up via Parallel Processing is Difficult.

Observation *A* (presented in section 5.1) suggests that the inherent parallelism available in expert systems is small. Observation *B* further suggests that:

i.) smaller production systems do not necessarily run faster than larger ones;

ii.) allocating one processing element to each RETE node (or production) is not a good idea because most of them will be idle most of the time; furthermore,

iii.) there is no reason to expect that larger production systems will exhibit more speed-up from parallelism.

Observation *C* suggests that scheduling is critical towards obtaining whatever (small) speed-up is available in the system. Unfortunately, dividing production systems into partitions which require similar amount of processing is difficult because good models are not available for estimating the processing required by productions and it varies over time.

Compile-time analysis/optimization on expert systems cannot be performed effectively because their run-time behavior is highly data dependent. An expert system contains a large body of knowledge capable of dealing with different situations. The actual situation to be tackled is not known until program execution time. Therefore, program behavior (such as frequency of procedure calls, amount of storage/communication requirements) is highly data dependent. Compile-time optimization techniques cannot be applied directly to such computations.

Synchronizations take place frequently in search problems. At the heart of many expert systems is a heuristic search problem: *given an initial state, apply knowledge to prune the search tree to arrive at the goal state.* This 2-phase cycle of knowledge application and problem state modification can be parallelized in many ways — each of which requires frequent synchronization. Consider the following examples:

- The RETE algorithm (OPS5): the *conflict-resolution* phase must complete before the *act* phase can begin. Even though the *conflict-resolution* phase could begin as soon as each rule successfully enters the *conflict set*, the *best* rule to be applied next cannot be determined until all candidates (including the slowest ones) have arrived.
- The Soar algorithm [25]: Computation is divided into an *elaboration* phase and a *decision* phase. Within each phase all productions satisfied may be fired concurrently. However, the *elaboration* phase must finish completely before the *decision* phase may proceed and vice versa.
- *And-parallelism* in Prolog: Common terms which occur in two clauses being worked on simultaneously must be share identical bindings. This requires tasks working concurrently on two sub-goals to communicate whenever such bindings are changed.

6. Conclusions

Many "building blocks" developed to enable parallel execution of expert systems have been surveyed in sections 3 and 4. Measurement results presented in section 5, however, seem to indicate that the inherent parallelism available in expert systems is small. Can expert system be formulation as highly parallel computations? Can these "building blocks" be put together effectively to support parallel computations? We do not have answers to these important questions. However, we would like to draw on some fundamental results concerning speed-up and parallel processing in section 6.1 and put forth some "fruit for thought" regarding future directions for research in section 6.2.

6.1 Speed-up and Parallel Processing

A small section of sequential code in an application can significantly limit its speed-up. Recall Amdahl's Law which states that the maximum speed-up S for a computation obtainable on a multiprocessor with **p** processors is governed by:

$$S \leq \frac{1}{f + (1-f)/p}$$

where **f** is the fraction of the computation that has to be executed sequentially. A simple application of this result suggests that parallel RETE-match algorithms can give at most a 10-fold improvement because only speeds-up the *match* phase which takes 90% of the execution time is affected.

When partitioning a single application into tasks, the *grain-size* of the tasks should be chosen such that: (i.) there is enough parallelism to exercise the PEs of the parallel processor and (ii.) communication and process management overhead must not outweigh the speed-up obtained from parallel processing. With production systems, it seems that extremely fine-grained tasks (of the order of 100 machine instructions) are needed for effective parallel execution [23]. Minimizing the scheduling overhead for such fine grain tasks is a major obstacle for achieving higher degree of speed-up.

A number of effective software organization structures have been proposed for multiprocessors. These include *software pipelines*, *systolic algorithms*, *divide-and-conquer* (*tree-of-processes*), and *relaxed* or *asynchronous processes* [26]. Speed-up could only be obtained, however, if certain criteria are met for each proposed organization. For example, *temporally decomposable* computations can also be arranged as software pipelines (which process data items incrementally from one stage to another). Processing at each stage may be carried out concurrently if data items can be *spatially decomposed* into (relatively independent) subsets. With *divide-and-conquer*, maximum speed-up is obtained when:

(i.) the *set-up* (task creation) and *trail-off* (recombination of results) times are small compared to the computation performed by each task;

(ii.) the number of tasks created is appropriate for the multiprocessor (given its task creation and management overhead); and

(iii.) tasks are effectively scheduled (mapped) onto the multiprocessor.

Whether an expert system can be spatially or temporally decomposed is application dependent. Decomposition boundaries can be identified based on a careful analysis of the nature of the input data-set and the reasoning process. Sometimes, these boundaries may not be obvious from first inspection. For example, KATE is an expert system for controlling the flow of conditioned air to maintain required temperatures, pressure and humidity levels within four compartments of the Space Shuttle while it resides in the Orbiter Modification and Refurbishment Facility at Kennedy Space Center. Parallelism can only be extracted by rethinking the problems KATE is trying to solve:

- monitoring sensors — data from different sensors can be processed in parallel;
- problem diagnosis — multiple fault theories and consistency checks can be pursued in parallel;
- control — alternative methods (i.e. set of commands required) for attaining a desired goal can be pursued in parallel; and
- multiple faults and complex control operations are spatially decomposable.

Researchers at the Intelligent Systems Technology Branch, Information Sciences Division of NASA's Ames Research Center are working towards a parallel version of KATE based on these dimensions of parallelism. Results should be available for publication next year.

6.2 Conclusions

WHAT IS THE BEST STRATEGY FOR BUILDING PARALLEL EXPERT SYSTEMS? Should we:

i.) *define a specific class of hardware architecture, then study the mapping of programs to these architectures* (e.g. **LISP* for the connection machine, *marker passing* on NETL, and *MultiLisp* for the BBN Butterfly)? or

ii.) *focus on a specific class of software architecture, construct a multiprocessor that best matches the program* (e.g. DADO for RETE, PIM for concurrent Prolog)? or

iii.) *establish a unified model to construct hardware and software architectures such that subsequent mapping between them can be easy and effective* (e.g. *CParaOPS5* for the Encore Multimax)?

We do not have an answer to this question yet. Nevertheless, we would like to suggest some research directions which seem most promising to us.

Requirements for parallel implementation should begin at the top of the software hierarchy and driven top-down — from problem solving paradigm design, to programming language implementation, to operating system, to machine architecture. We should decide the (macro) software organization most likely to extract parallelism from the knowledge-based application, before choosing *concurrent objects* vs. parallel Lisp, or shared-memory vs. distributed-memory architectures. In many cases, speeding up the "knowledge-based" portion itself may not produce the overall speed-up value we require. Bottom-up approaches produce machines that *could* exhibit orders of magnitude speed-up if *suitable* applications can be found.

The most efficient parallel execution model for expert systems may not look and work anything like the way they are specified. AI programming paradigms (whether it be *knowledge sources* with *blackboards* or *productions* on *working memory*) are designed to enable knowledge to be encoded and processed in a way similar to that carried out by human beings. They are not necessarily efficient for execution on a computer. However, when we stop asking "how computers can be modified to execute these paradigms directly", efficient execution models may follow. The RETE algorithm for sequential execution is a very good example.

In conclusion, we suggests that speed-up cannot come from parallelizing one particular existent paradigm or language or operating system. We must:

(i.) **understand how to break up the problem with minimal contention for accessing shared resources and reduced dependencies**; this could probably come about by considering (*macro* and *micro*) data dependencies in the system when designing its parallel implementation;

(ii.) **re-examine problem solving and representation schemes** (such as rules, blackboards, procedures, or logic programming) **and be open-minded about efficient parallel execution models that may not resemble the *human problem solving process***; and

(ii.) **explore parallelism at the application level**; the nature of the application may suggest temporal or spacial decompositions; do not the portion of the application that is not *knowledge-based* (e.g. re-organizing the I/O procedures may save more time than merely replacing the sequential inference engine with a parallel one).

References

1. David H. Warren, "An Abstract Prolog Instruction Set", Technical Note 309, Artificial Intelligence Center, SRI International, October 1983.
2. Tohru Moto-oka, Hidehio Tanaka, Hitoshi Aida, and Tsutomu Maruyama, "The Architecture of a Parallel Inference Engine - PIE -", Proceedings of the International Conference on Fifth Generation Computer Systems, pp. 479 - 488, 1984.
3. Salvatore J. Stolfo, Daniel P. Miranker, and David Elliot Shaw, "Architectures and Applications of DADO: A Large-Scale Parallel Computer for Artificial Intelligence", Columbia University, January 18, 1983.
4. Salvatore J. Stolfo and Daniel P. Miranker, "DADO: A Parallel Processor for Expert Systems", In IEEE, pp. 74 - 82, 1984.
5. Scott E. Fahlman, "Design Sketch For a Million-Element NETL Machine", Carnegie-Mellon University, Department of Computer Science, In AAAI-80, August 80.
6. James A. Hendler, *Integrating Marker-Passing and Problem-Solving: A Spreading Activation Approach to Improved Choice in Planning*, Department of Computer Science, The University of Maryland, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1988.
7. W. Daniel Hillis, *The Connection Machine*, The MIT Press, Cambridge, MA, 1985.
8. Atsuhiro Goto, "Research and Development of the Parallel Inference Machine in the Fifth Generation Computer System Project", Technical Report: TR-473, Institute for New Generation Computer Technology (ICOT), Minato-Ku, Tokyo, Japan, April 89.
9. K. Fuchi (ICOT, Japan) and M. Nivat (INRIA, France), editors, "Programming of Future Generation Computers", in the *Proceedings of the First Franco-Japanese Symposium on Programming of Future Generation Computers, Tokyo, Japan*, 6 - 8 October, 1986, Elsevier Science Publishers B.V., The Netherlands, 1988.
10. Richard P. Gabriel and John McCarthy, "Queue-Based MultiProcessor Lisp", Conference Record of the 1984 ACM Symposium on Lisp and Functional Programming, ACM, Austin, Texas, August 84.
11. Robert H. Halstead, Jr., "Parallel Symbolic Computer", Computer Magazine, 19:8, pp. 35 - 43, August 86.
12. R. Halstead, T. Anderson, R. Osborne, and T. Sterling, "Concept: Design of a Multiprocessor Development System", 13th International Symposium on Computer Architecture, Tokyo, pp. 40 - 48, June 86.
13. John S. Conery and Dennis F. Kibler, "Parallel Interpretation of Logic Programs", Communications of the ACM, May 81.
14. John S. Conery, *The AND/OR Process Model for Parallel Interpretation of Logic Programs*, Dissertation for Ph.D in Information and Computer Science at University of California at Irvine, University Microfilms International, Ann Arbor, Michigan, 1983.

15. M. V. Hermenegildo, "An Abstract Machine for Restricted AND-Parallel Execution of Logic Programs", University of Texas at Austin, Austin, Texas, 1985.

16. Doug DeGroot, "Restricted AND-Parallelism." Proceedings of the International Conference on Fifth Generation Computer Systems, OHMSHA, Tokyo, pp. 471 - 478, 1984.

17. M. Hermenegildo and Evan Tick, "Memory Performance of AND-parallel Prolog on Shared-Memory Architectures", Proceedings of the 1988 International Conference on Parallel Processing, August 15 - 19, 1988, Volume II Software, pp. 17-21, August 88.

18. Harold D. Brown, Eric Schoen and Bruce A. Delagi, "An Experiment in Knowledge-based Signal Understanding Using Parallel Architectures", Knowledge Systems Laboratory, Report Number KSL 86-69, Computer Science Department, Stanford University, October 86.

19. Bruce A. Delagi, Nakul P. Saraiya and Gregory T. Byrd, "LAMINA: CARE Applications Interface", Knowledge Systems Laboratory, Report Number KSL 86-67, Computer Science Department, Stanford University, November 87.

20. H. Penny Nii, Nelleke Aiello and James Rice, "Frameworks for Concurrent Problem Solving: A Report on Cage and Poligon", Knowledge Systems Laboratory, Report Number KSL 88-02, Computer Science Department, Stanford University, February 88.

21. James P. Rice, "Problems with Problem-Solving in Parallel: The Poligon System 1.0", Knowledge Systems Laboratory, Report Number KSL 88-04, Computer Science Department, Stanford University, January 88.

22. Charles L. Forgy, "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, 19: 17 - 37, 1982.

23. Anoop Gupta, "Parallelism in Production Systems", Ph.D Thesis, CMU-CS-86-122, Department of C.S., Carnegie-Mellon University, March 1986.

24. Leon Alkalaj, "Architectural Support for Concurrent Logic Programming Languages", PhD Thesis, Computer Science Department, University of California, Los Angeles, August 1989.

25. John E. Laird, *Soar User's Manual*, 4th Eidition, Xerox PARC, 1986.

26. Michael J. Quinn, Designing Efficient Algorithms for Parallel Computers. University of New Hampshire, McGraw-Hill Book Company, 1987.

N91-20701

THE ADVANCED SOFTWARE DEVELOPMENT WORKSTATION PROJECT

Ernest M. Fridge III and Charles L. Pitman
Software Technology Branch
NASA/JSC/PT4, Houston, TX 77058

ABSTRACT

The Advanced Software Development Workstation (ASDW) task is researching and developing the technologies required to support Computer Aided Software Engineering (CASE) with the emphasis on those advanced methods, tools, and processes that will be of benefit to support all of the National Aeronautics and Space Administration (NASA). Immediate goals are to provide research and prototype tools that will increase productivity, in the near term, in projects such as the Software Support Environment (SSE), the Space Station Control Center (SSCC), and the Flight Analysis and Design System (FADS) which will be used to support the Space Shuttle and the Space Station Freedom. Goals also include providing technology for future SSE and operational systems by adding knowledge based system support to all phases of information systems development, evolution, maintenance, and operation. The technologies under research and development in the ASDW project are targeted to provide productivity enhancements during the software life cycle phases of enterprise and information system modeling, requirements generation and analysis, system design and coding, and system use and maintenance. A programmable, Zachman-style framework is planned that will guide the information system modeling process and will be supported by system modeling tools integrated by a common knowledge base. An engineering graphical language will permit engineers to design applications and application templates. A software parts composition system will provide the environment for accessing parts, for "filling in the blanks" in generic parts, and for assembling parts based on the application templates. On-line user's guides will assist users in operating the developed information system with knowledge base expert assistance.

INTRODUCTION

Software development is a serious bottleneck in the construction of complex information systems, during both the development and evolution of such systems (Figure 1). Both development costs and maintenance costs can be high. The heaviest development costs tend to occur in the early part of the total life cycle during requirements generation, requirements analysis, design, and application development. Maintenance costs for sustaining the developed information system are even higher. An increase of the reuse of any of the software "parts" used in these activities has been viewed as a way to relieve this bottleneck. One approach to achieving software reusability is through the development and use of software parts composition systems [1,2].

A software parts composition system is a software development environment comprised of a parts description language for modeling parts and their interfaces, a catalog of existing parts, a composition editor that aids a user in the specification of a new application from existing parts, and a code generator that takes a specification and generates an implementation of a new application in a target language.

The Advanced Software Development Workstation (ASDW) is currently an expert system shell that provides the capabilities required to develop and manipulate these software parts composition systems. The ASDW is now in Beta testing at the Johnson Space Center. Future work centers on responding to user feedback for capability and usability enhancement, expanding the scope of the support for collecting, representing, and manipulating knowledge during the early phases of the information system life cycle (Figure 1), and in providing solutions for handling very large libraries of reusable components.

APPROACH

The ASDW is now moving into phase IV which will significantly broaden its scope of influence. Phase I (October, 1985 to April, 1987) demonstrated the feasibility of a knowledge based approach to application generation in a limited domain. Phase II (April, 1987 to February, 1989) investigated ways to exploit the use of knowledge representation, retrieval, and acquisition techniques. A prototype demonstrated a knowledge based system for the development of software parts composition systems (i.e., a software parts composition shell). Phase III (March, 1989 to December, 1989) prototyped ways to handle the scale-up problem (1000-100000 objects), prototyped ways to automatically generate the taxonomy, and initiated two Beta test projects at JSC. One test project is to generate trajectory mission planning simulations from software parts and the other is to provide expert assistance to operational users in setting up input data for simulations. The projects will support the SSCC and FADS, respectively.

During Phases II and III, the project also began joint activity with the United States Air Force (USAF), studying the information requirements of information systems, their integration and development processes, the methodologies required, and the integrated tools and integrated knowledge base that supports this development. The USAF is providing funding annually to study the modeling, methodologies, and information requirements of manufacturing information systems [3]. Modeling is based on the USAF's IDEF methodologies; IDEF is an acronym for ICAM (Integrated Computer Aided Manufacturing) Definition. JSC added funds to expedite the development of two methodology tools and a computer-assisted tutor to educate developers in the proper use of the methodologies.

Today, ASDW is the basic shell for supporting the reuse of stored information whether it be about software artifacts or any other design artifact. It contains object management and rule based constraint handling as well as a sophisticated "point and click" textual user interface (called "Specification-by-Reformulation") that models the way that people communicate among themselves. A neural net approach has been implemented to handle the large number of information objects that can be stored and a capability to automatically generate the taxonomy of objects has been incorporated. The "Help" system uses hypermedia technology.

Field testing by users in JSC's mission planning community was initiated in fiscal year 1989 (FY89) and has been a major thrust in FY90. Also, the user interface windowing system was made more portable with the migration to X-windows and TAE Plus [4]. In FY91, the total number of objects will be increased to handle a volume in the neighborhood of 100,000, and the user interface will be made more graphical with the capability to directly define application templates from a block diagram point of view. The modeling, generation, and reuse of early life cycle artifacts will be added along with the capability to use integrated methodologies and a common representation of their information content such that they can share common information.

In FY91, a programmable, Zachman-style framework [5] that will guide the knowledge acquisition and modeling process, the proper use of information system modeling tools, and the selection of the proper methodologies to be used, as a function of the characteristics of the information system, will be developed. This framework will be supported with an "Integrated Platform" of integrated services and a knowledge representation language that will integrate modeling tools used to define the information systems. These modeling tools will enforce the correct use of the methodologies.

Also during FY91, field testing will define techniques to support users in acquiring the knowledge required to operate the developed information system and will provide capability to assist users in the set up of operational data input with knowledge base expert assistance.

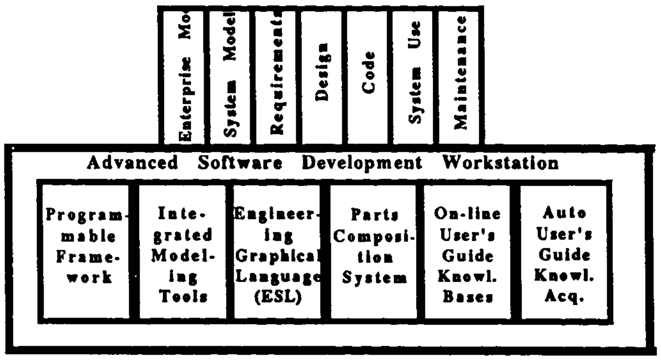
During Phase IV (which began January, 1990), CASE research and development will continue with the support of four organizations: Inference Corporation, SofTech, the Knowledge Based Systems Laboratory (KBSL) at Texas A&M University, and the National Research Council (NRC).

Inference will provide the reusable software library management system, and the user interface for accessing parts, for specifying new parts, and for assembling them to create applications. These capabilities will be incorporated into the component called Bauhaus [1,2].

SofTech will define an engineering graphical language called Engineering Script Language (ESL) which should significantly increase productivity in application generation. The ESL is a high level graphical language that permits engineers with a minimum of programming training to design applications that are populated with parts from the Bauhaus library. Constraint checking of the graph and the parts selection will be provided. The ESL is based upon proven concepts which have been put into operation at the Naval Research Laboratory in a restricted domain [6]. The ASDW effort will demonstrate its utility in the mission planning and analysis domain. The ESL will be integrated with the knowledge based library of reusable parts.

The KBSL will develop a programmable, Zachman-style framework [5] of information systems requirements that will aid in the use of methodologies (based on extended IDEF methodologies), integrated methodology tools, and the theory of information modeling (what models are required, when and how to produce them, how to use them, how to integrate their knowledge, etc). It will also aid in determining the activities for the various developer and user roles across the life cycle. All of the activities in the framework may be configured to match the characteristics of the information system being supported (e.g., business vs engineering). This on-line framework will be supported by and will help manage a prototype "Integrated Platform" which provides a uniform knowledge representation that will integrate methodologies and the models produced by them. This uniform representation of knowledge is the basis for integration and reuse of information among activities and information products throughout the life cycle of the information system.

Finally, the NRC will provide research into the use of expert systems to support the operational environment of the information system [7]. This study will develop a method to create a knowledge base, while an application is being developed, which can then help users easily set up complex input data in the completed application.



Enterprise Mo
System Model
Requirements
Design
Code
System Use
Maintenance
Advanced Software Development Workstation
Programmable Framework
Integrated Modeling Tools
Engineering Graphical Language (ESL)
Parts Composition System
On-line User's Guide Knowl. Bases
Auto User's Guide Knowl. Acq.

Figure 1 ASDW Project
The ASDW project consists of six research and development efforts aimed at supporting Computer Aided Software Engineering (CASE) throughout all phases of the software life cycle.

CONCLUSION

The CASE research and development being performed by the ASDW task is general in its scope and should benefit many NASA programs. The following sample benefits are anticipated for system developers, operators, and maintainers.

For information system developers, there should be an increase in productivity by providing application generation, parts composition, and an engineering graphical language. Reuse of all types of system development artifacts will be enhanced, and the knowledge existing in the various models of a system's phases will be integrated and translated through the life cycle. A goal is to provide integration of system modeling tools and advice on when and how to use the tools as a function of the type of information system to be developed; in particular, a goal is to provide an integrated knowledge base of the system's development which assists and guides the entire system development process.

For information system operators, the development of a prototype tool to create and maintain intelligent interfaces should increase productivity by providing on-line user's guides that include a hypertext help facility and a knowledge base to support data selection, to test for constraint violations, to generate input data streams and command streams from templates, and ultimately to supply expert advice on system operation, as a function of the operator's level of expertise.

For information system maintainers, all of the benefits to system developers will be available. Additionally, the integrated knowledge base built up automatically during an information system's development will capture the appropriate knowledge of the system's original developers and make this available to the system maintainers.

The ASDW project is currently field testing and refining the parts composition system and the on-line user's guide builder. Work is continuing on the engineering graphical language, the application template generation, the programmable framework, the integrated system modeling tools, and the integrated system knowledge base.

REFERENCES

1. Allen, Bradley and Lee, Daniel, "A Knowledge-based Environment For the Development Of Software Parts Composition Systems," ACM 0270-5257/89/0500/0104, pp. 104-112, 11th International Conference On Software Engineering, Pittsburgh, PA, May 15-18, 1989.

2. Allen, Bradley, "Case-based Reasoning In a Software Information System," submitted to AAAI-90/National Conference On Artificial Intelligence, Boston, MA, July 29 - Aug. 3, 1990.

3. Mayer, Richard, et. al., "Development Methodologies For Integrated Information Management Systems: Final Technical Report," Air Force Human Resources Laboratory/LRA, WPAFB, OH, July, 1988.

4. NASA Goddard Space Flight Center, "Transportable Applications Environment Plus," GSC-13275 with documentation, Computer Software Management and Information Center (COSMIC) at the University of Georgia, Athens, GA, Jan., 1990.

5. Zachman, John, "A Framework For Information Systems Architecture," G320-2785, IBM, Los Angeles, CA, March, 1986.

6. SofTech, Inc., "Graph Analysis And Design Technique Methodology," SofTech, Inc., Falls Church, VA, Nov., 1984.

7. Izygon, Michel, "Intelligent Interface For Complex Simulation Codes," Research Proposal to NASA/Johnson Space Center, National Research Council: Research Associateship Program, Washington, D.C., Sept., 1989.

P-CLIPS

Ross Miller
University of Lowell

(Paper not provided at publication date)

AUTHOR INDEX

REPORT DOCUMENTATION PAGE

1. Report No. NASA CP-3103, Vol. I	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Fourth Annual Workshop on Space Operations Applications and Research (SOAR '90)		5. Report Date January 1991
		6. Performing Organization Code PT4
7. Author(s) Robert T. Savely, Editor		8. Performing Organization Report No. S-618
9. Performing Organization Name and Address Lyndon B. Johnson Space Center Houston, TX 77058		10. Work Unit No.
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546 U.S. Air Force, Washington, D.C. 23304 University of New Mexico, Albuquerque, NM 87131		13. Type of Report and Period Covered Conference Publication
		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The papers presented at the Space Operations, Applications and Research (SOAR) Symposium, hosted by the Air Force Space Technology Center and held at Albuquerque, New Mexico, on June 26-28, 1990, are documented in these proceedings. Over 150 technical papers were presented at the Symposium, which was jointly sponsored by the Air Force and NASA Johnson Space Center. the technical areas included were: Automation and Robotics, Environmental Interactions, Human Factors, Intelligent Systems, and Life Sciences. NASA and Air Force programmatic overviews and panel sessions were also held in each technical area. These proceedings, along with the comments by technical area coordinators and session chairmen, will be used by the Space Operation Technology Subcommittee (SOTS) of the Air Force Systems Command and NASA Space Technology Interdependency Group (STIG) to assess the status of the technology, as well as the joint projects/activities in various technical areas. The Symposium proceedings include papers presented by experts from NASA, the Air Force, universities, and industries in various disciplines.

17. Key Words (Suggested by Author(s)) Life Sciences, Knowledge Acquisition, Biofilm Compass, Sharp/Robotic, Debris, Ultraviolet Radiation, CLIPS, Photometric, Neural Network, Space Station Freedom, Fuzzy Logic, Diagnostics/Algorithm, Microbiology, Ellipsometric, Biomedical	18 Distribution Statement Unclassified - Unlimited Subject Category: 59

19 Security Classification (of this report)	20 Security Classification (of this page)	21. No. of pages	22 Price
Unclassified	Unclassified	484	A21

*For sale by the National Technical Information Service, Springfield, Virginia 22161