SUPERVISORY AUTONOMOUS LOCAL-REMOTE CONTROL SYSTEM DESIGN: NEAR-TERM AND FAR-TERM APPLICATIONS

Wayne Zimmerman
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

Paul Backes
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

ABSTRACT

The JPL Supervisory Telerobotics Laboratory (STELER) has developed a unique local-remote robot control architecture which enables management of intermittent bus latencies and communication delays such as those expected for ground-remote operation of Space Station robotic systems via the TDRSS communication platform. At the local site, the operator updates the worksite world model using stereo video feedback and a model overlay/fitting algorithm which outputs the location and orientation of the object in free space. That information is relayed to the robot User Macro Interface (UMI) to enable programming of the robot control macros. The operator can then employ either manual teleoperation, shared control, or supervised autonomous control to manipulate the object under any degree of time-delay. The remote site performs the closed loop force/torque control, task monitoring, and reflex action. This paper describes the STELER local-remote robot control system, and further describes the near-term planned Space Station applications, along with potential far-term applications such as telescience, autonomous docking, and Lunar/Mars rovers.

INTRODUCTION

The original intent behind the design of the STELER local-remote robot control architecture was fourfold: 1) be responsive to recom- mendations coming out of NASA Space Station Extravehicular Activity (EVA) workload studies which strongly suggested ground-remote operations as a means of reducing EVA time and enhancing station utilization during the planned multi-year mantended phase (Refs 1, 2, and 3), 2) provide a modular design which would be more flight like (i.e., accommodate severe computational resource constraints at the remote robot site, manage either periodic remote site bus latencies (on the order of less than 1 sec) or manage major communication delays (greater than several seconds) between the ground and the remote site), 3) provide an easier means of transferring the technology to the flight centers, and 4) allow rapid tailoring or expansion of the system to other control applications. The current STELER local-remote design has evolved over a period of four (4) years.

Although the immediate application is Space Station telerobotics, the potential far-term applications include robot control/telescience between Earth and the Shuttle, Earth and the Space Station, and Earth and a rover on the Lunar or Mars surface. Other applications include autonomous docking, and terrestrial uses such as hazardous material handling, autonomous excavation, or undersea operations. Only the current planned work for space based applications will be discussed in this paper.

The Space Station robot control environment provides a good example of the intended application environment as a means of setting the stage for discussion of the local-remote control system design. In order to perform ground-remote operations the control system will have to manage communication delays on the order of 8 to 9 seconds (Ref 4). The majority of this delay is in the processing/packaging and relaying of the data from the ground, through TDRSS, then to the Space Station communication subsystem/operator workstation, and finally to the robot controller. To manage this time delay and provide robust control with limited on-orbit computational resources, the JPL STELER local-remote control architecture off-loads most of the high level work envelope, task assessment and planning functions to the local site. The local site uses low bandwidth stereo video images captured at the remote site from the robot/workcell cameras as the initial means of determining the actual position of objects in the environment (i.e., are objects where they are supposed to be, what is the viewing angle/ lighting environment like). The stereo video is displayed on a Silicon Graphics Inc. (SGI) workstation which also allows wire-frame, or solid, graphic models of objects in the workcell to be overlayed onto the video to verify the geometric model on the scene. The local site operator matches the graphic overlay with the video to determine the object's location/orientation and the world model is updated. This information is provided to the User Macro Interface (UMI) residing in the SGI where it is now used to parameterize the robot control macros to manipulate the object. The operator assembles the desired task control macros into a sequence and then sends them to the remote site Modular Telerobot Task Execution System (MOTES) where the commands are then executed and monitored at the robot site. MOTES is functionally equivalent to a spacecraft control system (e.g., Galileo). A command interpreter and dispatcher respectively manage incoming/ outgoing data. Sensor and Monitor modules process robot sensor data and check sensor thresholds/fault conditions. The Control module contains the trajectory generator and position based impedance control functions and generates the task space robot motion. A Fusion module merges motion commands from the various command sources (e.g., trajectory generator, hand controller commands, force sensor). A Device Driver module provides the low level communication to the physical components such as the robot arm and gripper. The next section provides a detailed discussion of the above system design elements and also describes the current/future laboratory hardware implementation.

LOCAL-REMOTE SYSTEM DESIGN

The above introduction briefly described the functions of the primary components in the local-remote robot control architecture. To lay the foundation for understanding planned applications, a more detailed system description is needed. This design description is provided in the following paragraphs.

A. Local Site System Design

The local site subsystem design has several components (Refs 5, 6) **Figure (1)** provides a schematic of the software modules which reside in the local site. **Figure (2)** shows the current laboratory operator workstation. Starting with the

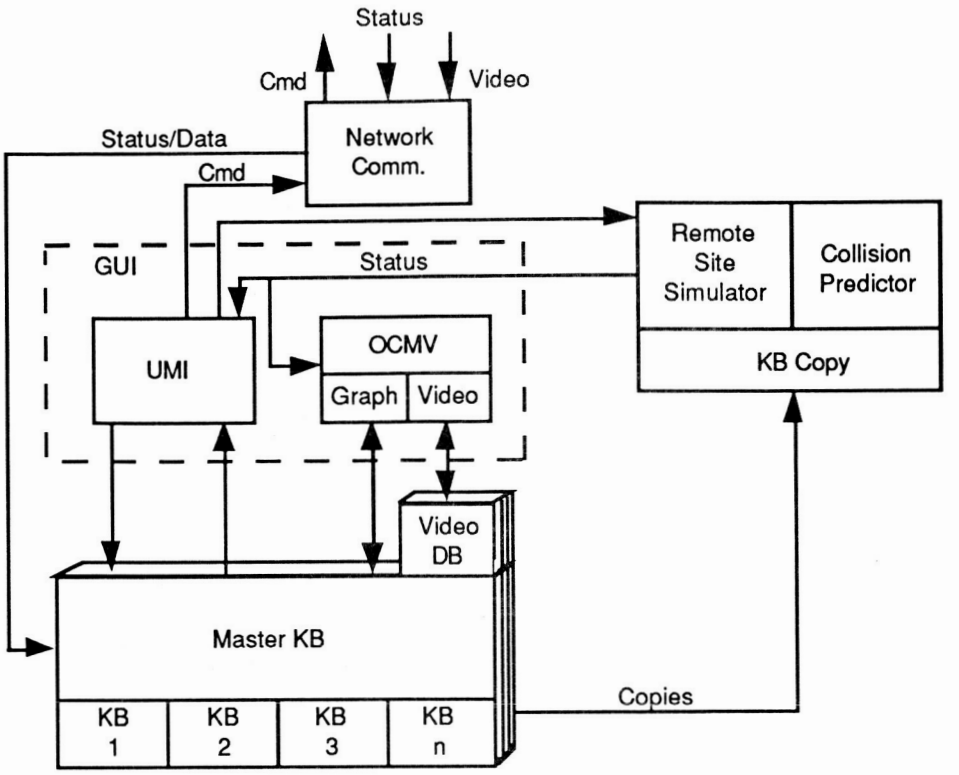

Figure (1) Local-Site Design

Figure (2) Current Laboratory Operator Workstation

operator interface, the operator accesses the various high level control menus and database through the keyboard, mouse, and spaceball input devices. The robot hand controller is the input device for controlling the manipulator(s) with tele-operation, and currently has a separate communication link to the MOTES subsystem at the remote site.

Moving to the next level, a **network comm** module with time delay buffer serves as the gateway for outgoing/incoming data between the local and remote sites, and for imposing variable time delay on return traffic. The hub of the local site revolves around the **user macro interface (UMI)** (manipulation) and **operator coached machine vision (OCMV)** (perception) modules, which reside within a high level **graphical user interface (GUI)** structure, and the **master knowledge base (MKB).** The local site presents to the operator a window-based GUI for commanding and sequence editing; and, a video/graphic interface that allows the operator to effectively perceive the task environment and to interactively modify the MKB. The GUI has four standard manipulation windows which are always visible-sequence, command, status/environment, and perception. The sequence window displays previously built sequences which are available. The command window displays the commands in the current sequence, and allows the operator to edit and create commands. Within the command window, a macro window displays the currently active control command/macro. The status/environment window displays the current system state information (e.g., which arm is active, is the system in "simulate" or "real" mode). The perception window provides the operator with options for updating video, graphics overlay, and changing the eye point of view. The GUI keeps track of, and displays, the current operation being performed and the most recent values of environment variables. **Figure (3)** shows a typical GUI display.

The UMI module is used to design, build, execute, store, and replay manipulation commands to be sent to the remote site for execution. The operator can interactively build/simulate complete task sequences using the menu interface provided by UMI. UMI provides parameter input checking, menus of robot control macros, sequence building, sequence simulation/execution, and status display.

OCMV contains both graphics and video elements. Using a menu interface, the operator can command video capture, thus allowing update of the local site video image of the workcell at the remote site. Once an updated set of images is returned and displayed, the operator can selectively relocate objects in the workcell using the graphic overlay capability. This is done by using either stored object models of known objects (called in from the knowledge base by clicking on the desired object in the video display with a cursor), or by performing an object build/edit on-line. The operator overlays and matches the object model with the video image using the spaceball. Once the overlay is properly mapped onto the video image, the machine vision object localization module determines the object's position/orientation using the internal camera models (e.g., viewing angle, focal point(s)) and position of the cameras relative to the workspace/object. The OCMV software also maintains known camera arm positions to allow different views of the workcell/

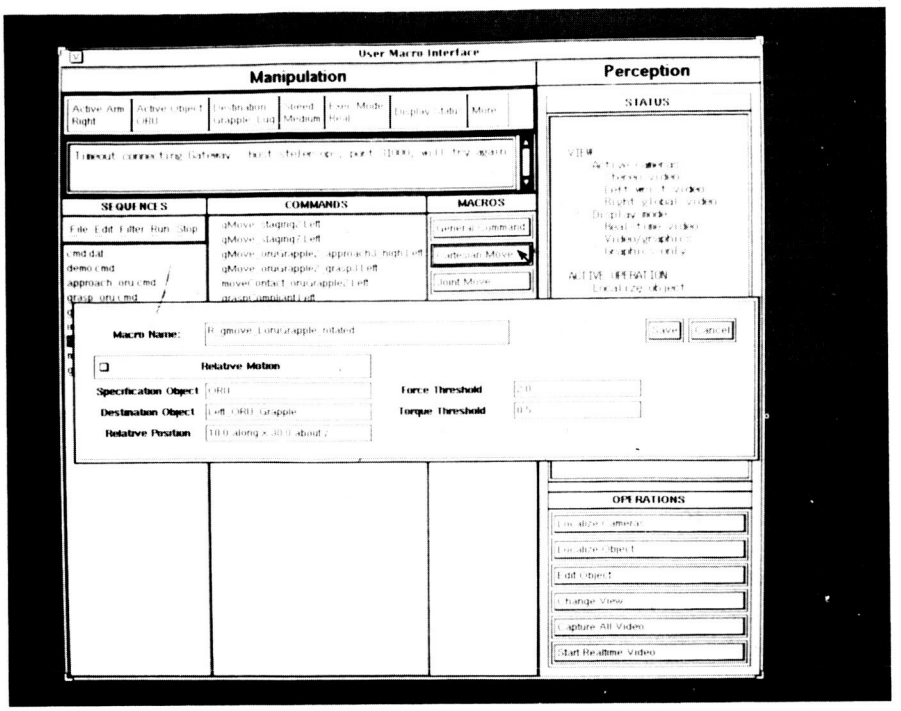

Figure (3) Typical GUI Display

object. **Figures (4) and (5)** display the graphics and video overlay capabilities respectively.

The last major component, the MKB module, is made up of nodes and links. Nodes represent real or fictitious objects; and, links represent relationships between the objects. A node contains object properties(e.g., polyhedral description, name, mass) and a link contains data about a relationship (e.g., object connectivity, coordinate transformations). The relationships between objects, either rigid or movable/breakable, are organized into parent-child links. The MKB also containsa master kinematic database, control/ dynamic parameters, simulator database (with collision prediction data), video database, and graphics database. The MKB also manages data storage/retrieval. The operator has the responsibility to maintain consistency within the MKB. This means that the MKB does not have an internal expert or rule based system which automatically checks and maintains consistency.

Figure (1) shows one more major component, the **remote site simulator and collision predictor (RSSCP).** In the near term, the remote site simulator located in MOTES will be called from the local site to test a task sequence before actual execution. The status will be returned to the local site and viewed with the graphics simulation. Eventually, a copy of the remote site simulator will reside at the local site and will simulate not only the system state transitions, but also the kinematics and simple force contact models (e.g., magnitude but not vectors). The

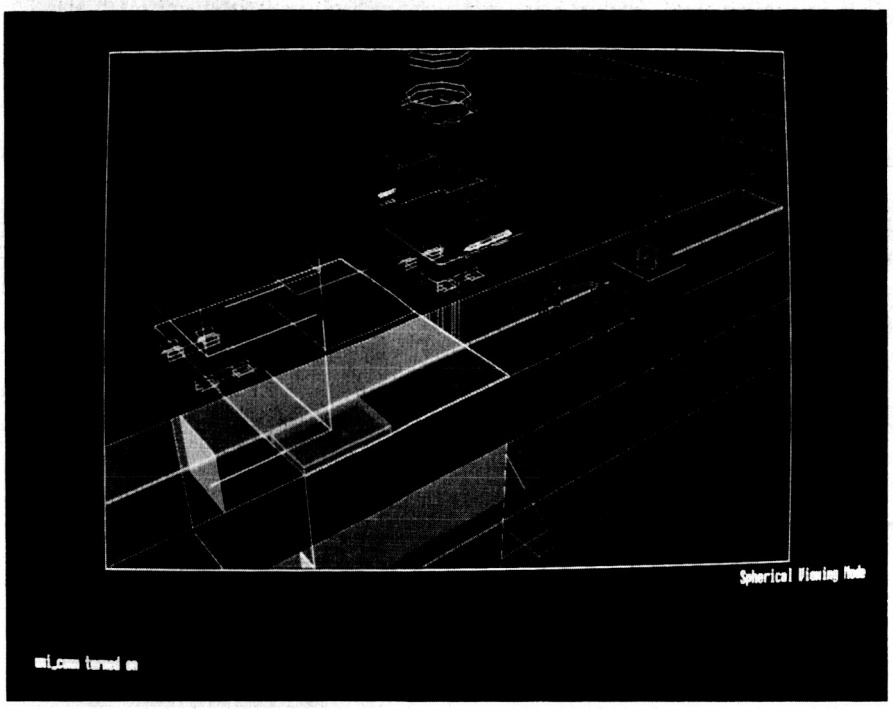

Figure (4) Perception Graphics Overlay

Figure (5) Perception Graphics Overlay on Video

collision predictor takes trajectory generator via points, arm kinematics, and global geometry and checks for collisions with objects in proximity to the joints as the trajectory is executed in the simulator. Collisions are highlighted and an eventual safe trajectory is iteratively generated.

B. Remote Site Modular Telerobot Task Execution System (MOTES) Design

The MOTES robot control design has been discussed in detail in other literature (Ref 7), and will therefore, only be summarized here. As stated above in the Introduction, MOTES has several modules. Each module interfaces to the rest of the system through shared memory with specified input and output parameters/functionality. **Figure (6)** provides a functional diagram of MOTES. The various modules operate asynchronously with the **executive** module handling communication with the local site, placing new commands into the task/

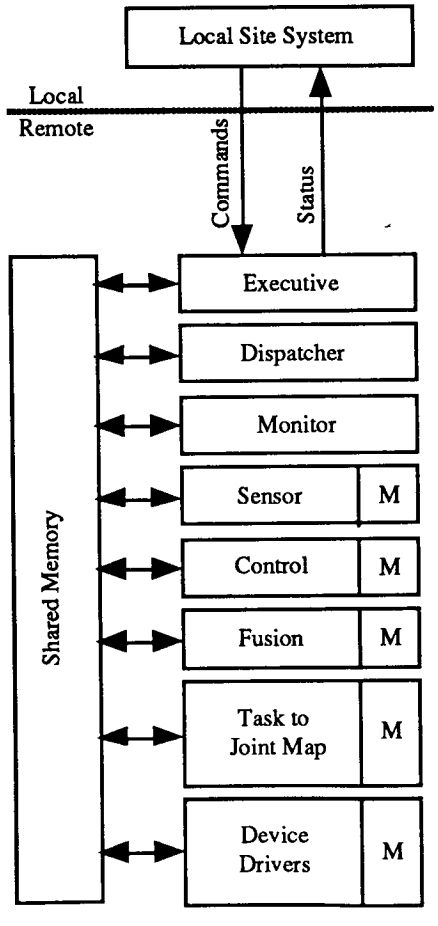

Figure (6) MOTES Remote Site Functional Diagram

reflex command queues, and returning status data to the local site. The **dispatcher** module checks/controls the transition between execution states by monitoring the status of the various modules and specifying the appropriate commands/parameters to the various modules via shared memory. The dispatcher controls the transition between commands by changing the active command block parameters as specified by a monitor event (e.g., completion of a command sequence, setting of an error flag causing a halt in command execution). If an event terminates under an acceptable condition (e.g., trajectory successfully executed), then the next command in the queue is issued. If an unacceptable termination condition results, then the dispatcher finds which reflex command queue to use based on the matching flag set in the reflex table associated with that event. The dispatcher executes the set of reflex commands, clears the task command queue, and sends a report back to the exec stating which reflex action was initiated.The **monitor** module(s) check the behavior of the system and set the appropriate success/error flags when a monitor event has occurred. Examples of monitor events were provided in the preceding paragraph. Monitoring occurs at two different levels: 1) the system level (e.g., multi-sensor based monitoring, command completion), and 2) the sensor/reflex level (e.g., force threshold exceeded). In designing the system, previous experience dictated that it is important to monitor and control low level events where the information is generated and the context is known. This design also improves response time for critical events.

The **sensor** module(s) generates either real (e.g., force/torque sensors, encoders, proximity sensors), or virtual (e.g., bounds on distances to joint limits, repelling forces when within a given distance from collision points), sensor data which is placed in shared memory for use by the other MOTES modules.

The **control** module(s) generates manipulator setpoints, and performs control based on both the local site generated command sequence and sensor data. Each control module has a uniquely specified Cartesian control frame for control of the manipulator. The complete motion command specified by each command module is transformed to a common task space (consisting of the common Cartesian frame and arm angle) before being placed in shared memory. The position trajectory generator is the control module which computes the desired task space path including Cartesian frame and arm angle.

The **fusion** module combines the commands from the various control modules as specified in the task parameters sent by UMI. For sensor based control, the trajectory generator employs a position based impedance model in each degree of freedom of the task space. The fusion module also allows position based inputs from the teleoperator to be fused with autonomous sensor based control inputs (provided by the monitor/control modules). The subsequent **task-to-joint map** module maps the task space command of the fusion module to the actuator space of the physical device being controlled.

Last, the **device driver** module is responsible for communicating commands, and status, with the actual system hardware as well as performing hardware specific

computations. The primary device drivers are for the manipulator (Robotics Research Arm), force/ torque sensor (Lord), and servoed gripper (Telerobotics Research Inc. (TRI)).

C. Current/Future System Implementation

To set the final stage for discussing applications, it is important to understand the current hardware/software implementation envir-onments. The local site system currently runs predominantly on an SGI 310/VGX workstation, using the IRIX 4.0.4 operating system. The SGI workstation was selected because the target Space Station Mission Operations Planning environment at Johnson Space Center (JSC) for telerobotics is the SGI. A Sun 4/260 workstation at the local site is used as to develop and test the remote site MOTES real-time Ada code. As stated earlier, in the near term the remote site simulator running in MOTES will be used to simulate task sequences. However, it is planned to eventually test MOTES and simulate task sequences using a carefully descoped version of MOTES running on the Sun 4 (i.e., the sequence error checking, trajectory generator, and position success/error flags are the only predominant features needed for local site simulation). In the future, it is desirable to port all local site software to the SGI. To round out the local site hardware, a 6DOF JPL/Salisbury Model C hand-controller and motor controller and associated VME chassis allow teleoperation of the 7DOF arm. An indexing trigger on the hand controller provides the additional means for controlling the redundancy, or seventh degree of freedom. MOTES is written in Ada and runs in a VME environment on Heurikon 68020 processors. A Verdix/ VADSworks Ada cross compiler is used to produce the target code. The target code runs on top of the VADS-works operating system in the 68020 environment. The Verdix/VADSworks Ada compiler and 68020 environments were selected primarily because they provided the desired performance for the least investment. Additionally, since the Space Station software environment will be Ada, it was decided that by moving to real-time Ada the system would be more in synch with the planned flight software for the station remote manipulator and the Canadian Special Purpose Dexterous Manipulator (SPDM). Communication between the 7DOF Robotics Research arm (model K1207) controller and the MOTES VME chassis is through a pair of Bit-3 memory interface cards. The direct servo control of the arm joints is done through the factory supplied controller.

Since the 68020 CPU's used in this initial version of MOTES are relatively slow (the 68020 V2FA runs at approximately 1.7 MIPS), a multiple CPU environment was required. This decision to go to a multi-processor environment was also considered viable because the SPDM design is currently a multi-processor configuration. Therefore, it was considered useful to explore and resolve potential scheduling/timing problems associated with real-time control in a multi-processor asynchronous environment. Figure (7) displays the current VME multi-processor environment running in the lab. Although the system is operational, based on current test results, it appears that it is desirable to reduce the number of multi-process-or boards from the present five to one, or two. To achieve this, the new MOTES design in considering an 80386 configuration, running the commercially available Lynx operating system. This configuration will condense MOTES, increase performance, and also provide a compact flight qualifiable controller for

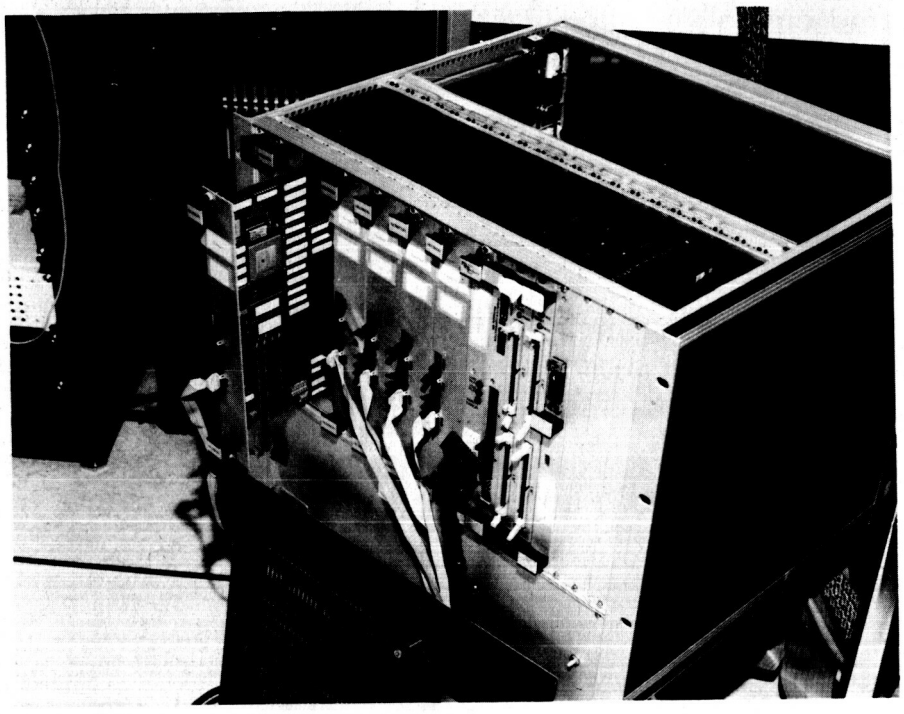

Figure (7) STELER MOTES Remote Site Controller Hardware

systems. As will be seen in the following discussion, the planned evolutions of both the local site (full SGI implementation) and remote site (80386 flight qualifiable configuration) will be important to achieving the planned applications.

PLANNED APPLICATIONS

A. Near Term

The current JPL STELER local-remote control system has several immediate applications relative to the Space Station flight program. First, before the station is ever launched it is important to start the process of assembling and testing robot control sequences for station/payload servicing now. These sequences must be tested on the ground under high fidelity task simulation conditions in a laboratory (e.g., lighting/lighting angle, object/shadow occlusions). Further, to the degree possible, similar robotic devices (SSRMS/SPDM) and control environments should be used to duplicate operator/system control response. Although the degree to which the U.S. NASA community is planning to perform task verification is still uncertain (i.e., partial verification using graphics simulation vs. full task verification in a laboratory with real robotic/task hardware), it is still important to have the technology ready soon to enable start of task sequence assembly/testing. The JPL system is currently on schedule for delivery into the JSC Space Station ground test/evaluation laboratory by the end of FY93. As a spin-off from the above ground

application, the system (which allows either full manual teleoperation of the robot(s), shared control, or supervised autonomy) also provides an essential training medium for astronauts from the standpoint of both hands-on control, and assessing human-machine tradeoffs in the performance of various tasks under varying environmental conditions. For example, the astronaut operators may determine that for some manipulation tasks in complex/occluded environments, it is more efficient/safer to have the operator perform the gross motion control while allowing the autonomous system to set the arm pose and perform the proximity/fine alignment control.

The most powerful application of the STELER local-remote robot control system is for ground control of on-orbit station robots. The recent mid-year review of the JPL project status, by NASA headquarters, revealed that the downscoping of the station (and subsequent reduction in astronaut crew) has still not resolved the substantial projected on-orbit servicing/payload workload (Ref 8). As a result, the most logical way to reduce on-orbit workload and increase station utilization (i.e., in the early man-tended phase of operation, the Space Station will not be permanently manned and will therefore see an estimated utilization on the order of 13%) is to perform simple setup and inspection tasks from the ground. To this end, the local-remote control system has been tested in actual hardware/software segregation scenarios such as 1) the control of an industrial robot inspecting a Shuttle payload rocket booster at Kennedy Space Center, from JPL's STELER lab located 3000 miles away using an early local-remote architecture functionally equivalent to the current design (time delay of 2-3 seconds) (Ref 9), 2) control of the JPL STELER robots from a temporary workstation located in a trailor approximately 100 yards away (imposed time delay ranging from 0-9 seconds) (Ref 10), and 3) control of a PUMA robot simulation located at Texas A&M University, from JPL, via the Internet/TelRIP communication network (time delay of roughly 1-3 seconds) (Ref 11). It is also planned to control the JSC Robotics Research arm from JPL as well, and use the Internet/TelRIP interface to both send and debug software used in the JSC telerobot test/integration laboratory. Again, by streamlining the hardware/ software architecture, the transfer of this technology will be greatly simplified.

The last major near-term application of the JPL local-remote architecture is somewhat related to the earlier ground-based system testing/training, but applies to on-orbit robot operations. Currently, it is planned to control the station robotic systems primarily in teleoperation mode. However, due to limited camera views, potential lighting/occlusion problems, and, most importantly, limited on-board computational resources, teleoperators will need to depend greatly on ground task simulation/verification before proceeding. The STELER local site on the ground will provide a means for accurately updating the ground workcell model even in the presence of limited viewing/occlusions. The local-remote site combined will provide a high-fidelity simulation of the task, and assembly of the ultimate control macros which get telemetered up to the station teleoperator workstation for the astronaut to use during task execution as a control template.

B. Far Term

As stated in the "implementation" subsection, it is planned to move to more "flight like" versions of the local-remote site systems by implementing the local site completely on a SGI workstation, and the remote site on an 80386 computational environment. Work just being initiated for FY93 in STELER calls for an expansion into the new area of telescience. New work will start on the design and construction of a "mini-device" (called the Lab Tending Telerobot (LATT)) which can be placed as a module on the front of a science rack, or inside, and which can be programmed/controlled from the ground by the principal investigator (PI). The device will allow the PI to physically interact with the science experiment(s) on-board the Shuttle (intermediate term target), and eventually the Space Station or free flying platforms (far term target). The more compact hardware/software implementation will allow LATT ground and on-orbit elements to be respectively, 1) easily ported to the PI's base facility, and 2) more easily space qualified since the 80386 Intel processor has already been space qualified.

Another more intermediate-to-far term application is for autonomous docking. As a test of the STELER control system robustness, an autonomous docking task has been structured. The female fixture is mounted on a full scale satellite mockup which hangs freely from a cable and has five degrees of freedom. The male fixture is mountable on the 7DOF arm, or on a jig fixture. The control scenario calls for the local site to update the position of the fixture on the satellite, followed by either the manipulator acting as the chase vehicle and autonomously docking with the satellite; or, the arm grasping a grapple fixture attached to the satellite and moving the satellite to dock with the jigged male half. Work planned for FY93 calls for the STELER team to begin building the appropriate graphics, simulation, and control macros for an autonomous docking workstation running on the SGI, with eventual transfer to the Marshall Space Flight Center (MSFC) docking simulation facility.

The last long term application of the local-remote control system is for Lunar/Mars vehicles. For potential early precursor missions, either autonomous mini or micro-rovers may be used to map potential landing sites and perform science experiments (e.g., soil sampling, coring, experiment module placement). With time delays on the order of many seconds (Lunar) to many minutes (Mars) it is imperative that a local-remote control architecture like STELER's be used to insure robust/safe control of the robotic vehicles and manipulation functions. The current move towards a more efficient, compact remote site controller design is the next step towards learning how to provide robust remote site control/intelligence within tightly constrained volume/mass requirements.

CONCLUSIONS

A description of the current JPL STELER local-remote telerobot control system was provided along with present and future hardware/software architecture implementations. Additionally, a discussion of both near and far term applications was provided and mapped onto the planned hardware/software evolutions. Most

importantly, this paper provided a roadmap and structure on how the STELER team plans to carefully evolve, test, and implement the local-remote control capability towards an expanding application domain.

ACKNOWLEDGMENTS

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

1. Zimmerman, W., Swietek, G., "Factors Pointing to Evolution, A&R," NASA, 1989 (Informal document).
2. Weeks, D., Zimmerman, W., Swietek, G., "Space Station Freedom Automation and Robotics: An Assessment of the Potential for Increased Productivity," NASA/MITRE, December 1989.
3. Fisher, W., Price, C., "Space Station Freedom External Maintenance Task Team Final Report," Vols. 1 and 2, NASA, Johnson Space Flight Center, July 1990.
4. Aster, R, Pitahaya, J., Deshpande, G., "Analysis of End-to-End Information System Latency for Space Station Freedom," JPL, Doc. no. D-8650, May 1991 (Informal Document).
5. Beahan, J., "Local Site Software Design," JPL, June, 1992 (Informal Document).
6. Bon, B., Beahan, J., "A Graphics Based Operator Control Station for Local-Remote Telerobotics," SPIE, April 1992.
7. Backes, P., Long, M., Steele, R., "Designing Minimal Space Telerobotics Systems for Maximum Performance," AIAA, February 1992.
8. Zimmerman, W., "JPL Space Station Engineering Prototype Development (EPD) Program: Telerobotics Mid-Year Review, JPL, NASA HQ Level I/II, June 1992 (Presentation).
9. Bon, B., et. al., "Telerobotic Control from 3000 Miles," JPL/NASA Kennedy Space Center, November 1989 (Laboratory Demonstration).
10. Zimmerman, W., Backes, P., Bon, B., Long, M., Steele, R., "JPL FY91 End-of-Year Telerobot Control Review," November 1991 (Laboratory Demonstration).
11. Tso, K., "Remote Control of Texas A&M PUMA-Simulation," March, 1992 (Laboratory Demonstration).