The JPL Telerobot Manipulator Control and Mechanization Subsystem (MCM)

S. Hayati, T. Lee, K. Tso, P. Backes, E. Kan
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California, U.S.A.

J. Lloyd, McGill University, Montréal, Québec, Canada

Abstract

The Manipulator Control and Mechanization (MCM) subsystem of the JPL/NASA Telerobot system provides the real-time control of the robot manipulators in autonomous and teleoperated modes and real-time I/O for a variety of sensors and actuators. Substantial hardware and software are included in this subsystem which interfaces in the hierarchy of the Telerobot System with the other subsystems. These other subsystems are Run Time Control (RTC), Task Planning and Reasoning (TPR), Sensing and Perception (S&P), and Operator Control subsystem (OCS) (see Figure 1). This paper describes the architecture of the MCM subsystem, its capabilities, and details of various hardware and software elements. Important improvements in the MCM subsystem over the first version [1] are dual arm coordinated trajectory generation and control, addition of integrated teleoperation, shared control capability, replacement of the Unimate controllers with JPL-built motor controllers [2], and substantial increase in real-time processing capability.

1 Introduction

The Jet Propulsion Laboratory, National Aeronautics and Space Administration (JPL/NASA) telerobot program began in 1985 [3]. The goal of this program is to develop a telerobot system that has the capability to perform typical space-related robotics activities under human supervision. It is understood that the system will initially have more capabilities in teleoperation and as it develops, more autonomous capabilities will be realized. This telerobot system consists of five subsystems [4]. Since each subsystem is specialized in one particular area of robotics, and each subsystem will in general evolve based on current technology, no rigid guidelines were imposed on the subsystems to utilize particular hardware or software. It was determined in the early stages that as long as all the subsystems could communicate with each other on a common protocol, then they could develop at their own pace.

Figure 1 shows a schematic drawing of the overall system. Two paths from the operator to the robots are recognizable in this figure: the Teleoperation path and the Autonomous path. In

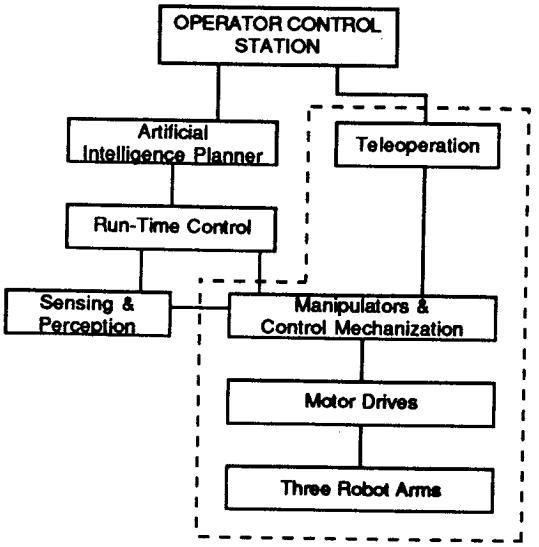

Figure 1: JPL System Architecture

the Teleoperation path, the operator controls the robot arms using the six-degree-of-freedom force reflecting hand controllers (FRHC) [5]; minimal interaction with other subsystems is required. In the autonomous path, the operator can perform supervisory control by issuing high level commands via the TPR and the RTC subsystems to invoke certain general *macros* resident in the MCM subsystem.

The scenario of the operation of this telerobot system is a 'mixed' mode of operation. As an example, assume that the operator needs to perform a bolt removal. She tells the TPR subsystem that she needs to operate one of the arms in the teleoperation mode. This command is then channeled through the RTC subsystem to the MCM subsystem instructing it to operate in teleoperation mode. As a result of this action, now both the TPR and RTC are aware that the environment and the position of the robots might change. They "listen" to the MCM subsystem as the teleoperation continues so that they can "watch" over the shoulders of the operator making sure that she does not move the arms to some forbidden area based on their a priori knowledge of the environment. She then uses the TPR subsystem to request the S&P subsystem to locate the position of the bolt using a designation and fit mechanism [7,8]. This process enables the operator to locate the position of the bolt head to within a few mm. At this time the operator can request the RTC subsystem to initiate a relative calibration process with the S&P subsystem. The S&P subsystem then uses the *camera arm* to look at the robot end-effector and the bolthead to locate more precisely their relative displacement. This process eliminates the effects of the absolute positioning errors of the manipulator arms. The operator then issues a high level command for bolt removal. The TPR subsystem breaks this command into several subcommands which are sent to the RTC subsystem and RTC issues appropriate commands to the MCM subsystem to position the arm above the bolt and then to execute the MCM bolt removal macro. MCM macros are written in a general way so one macro can be used for a number of geometrically similar objects and tasks. The RTC subsystem also checks all possible joint limit violations before it sends a command to the MCM subsystem. MCM removes the bolt and reports the success or failure status to the higher level systems. The MCM subsystem is responsible for providing the following basic manipulation functions:

- autonomous and teleoperated control of two arms,
- force reflection to the hand controller and display to the operator,
- execution of free or guarded motions,
- execution of macro tasks capable of accepting different parameter sets,
- frequent trading of control between the teleoperated and autonomous task executions, and
- shared control.

In section 2 we provide the basic architecture of the MCM subsystem. Section 3 describes the new multi-arm RCCL and its port to a Sun 4/200 computer. Section 4 details the teleoperation and shared control portions of the system and conclusions are given in section 5.

2 MCM Architecture

The MCM subsystem is a real-time system which controls the robots and the hand controllers, acquires sensor data for control and communicates with other subsystems. Guidelines for the design of this subsystem are as follows:

- MCM's computational architecture must clearly distinguish between the *local* and *remote* sites. The *local* is where the human operates the system and *remote* is where the manipulators operate. In principle, these two sites can be only a few feet or hundreds of miles apart.
- The architecture should allow for future expansions of the system's computing power and addition of more peripherals.
- The architecture must use the same remote computational environment to operate the system in either teleoperation, autonomous, or shared control modes.
- System software must be flexible to add/modify new basic functionalities such as a user-defined trajectory generator. This feature is required to design shared control [6] capabilities so the operator and the autonomous system can *share* in the execution of various tasks.
- The software must be as arm independent as possible.
- The system must allow for rapid software development both at the system and user levels.
- The computing environment (i.e., hardware and software) must allow the system to migrate to new faster processors as they become available without major rewrite of the software.

Figure 2 shows a block diagram representation of the MCM subsystem. The system is physically divided into two sites: local and remote. In the local site where the operator is stationed, there are two identical systems (left and right) each consisting of a hand controller (FRHC), a VME chassis with 68020 processors for computations of kinematics and force reflection, an RGB monitor for force/torque display, a JPL-built Universal Motor Controller (UMC), and a Sun computer for the

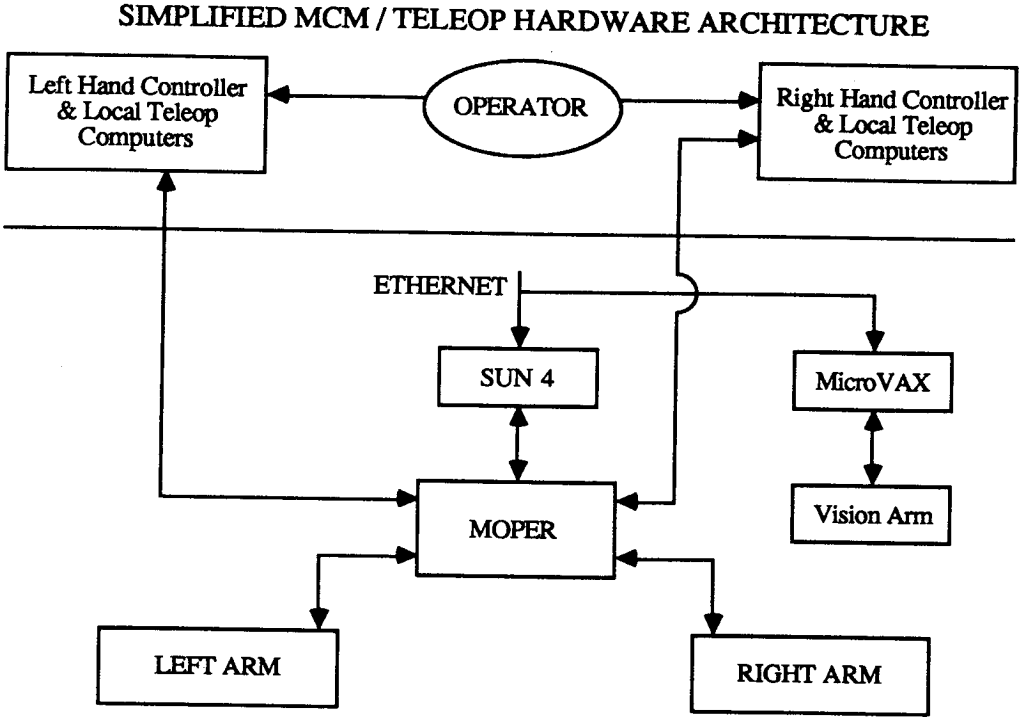

Figure 2: High-Level MCM Architecture

operator interface. The remote site consists of a Sun 4/200 computer to run the enhanced version of multi-arm RCCL [9,10,11] which is capable of generating trajectories for multiple coordinated manipulators. This is used to run two PUMA 560's. A MicroVax II which runs the same enhanced RCCL is used to move a third Puma 560 for pointing a pair of cameras for the S&P subsystem. The reason for not using the Sun 4 to run this third arm is the unavailability of one additional UMC rather than any limitation in the new Sun 4-based RCCL software. Each arm is equipped with a Lord wrist force/torque sensor, servoed gripper from TRI Inc., and the Unimation teach pendant which operates under RCCL. Signals from local hand controllers, force/torque sensors, servoed grippers, and teach pendant all are read and passed to the Sun 4 through the MOPER. This chassis is equipped with four processors[1] and several parallel and serial communication cards. Figure 3 shows a block diagram representation of the MCM hardware and their connections.

2.1 Real Time Operations

The system works based on interrupts generated by a processor card in the MOPER chassis. This regular interrupt causes the MOPER to gather information from the UMC's (i.e., the state of the arms such as joint encoder values, joint potentiometers, and the status of the motor power), the incremental motion of the hand controllers (either in Cartesian or joint form), and force/torque sensor information. This data is then transmitted to the SUN 4/200 via a shared memory card (BIT 3). This data is used in RCCL to check for joint or velocity or some other user-defined limit violations.

[1] All processor cards except the ones in the UMC's are single board computers base on Motorola 68020 microprocessor and 68881 floating point co-processor with 1 meg. RAM. The development environment is VxWorks with the Wind kernel.

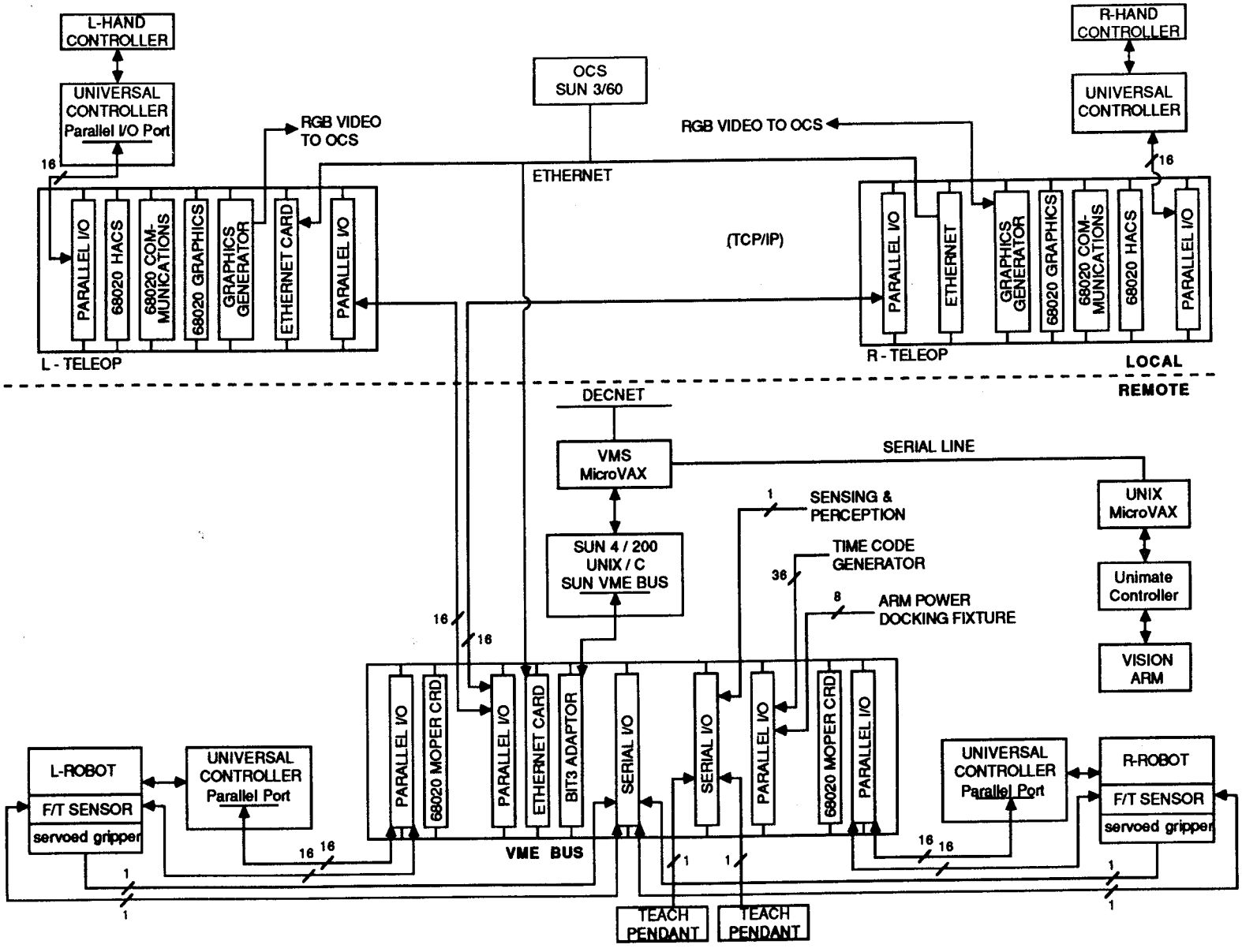

Figure 3: MCM Hardware Block Diagram

Since RCCL allows the trajectories to be modified in real time, this data is utilized to modify the nominal trajectory to provide sensory-based motion. During this time interval, MOPER transmits force/torque sensor information to the local teleop chassis for Kinesthetic and visual feedback to the operator. It also receives commands from the Sun 4 and after interpretation sends appropriate commands to the UMC's to move the robot arms. In addition to the I/O function, MOPER generates the main timing interrupts to the system. This means the Sun 4, which runs the RCCL code as well as the local teleoperation chassis, operates in the slave mode. This system heartbeat can be increased or decreased based on the complexity of the user written software in RCCL. Unlike the Unimate controller restriction where one is limited to sampling time intervals of 7, 14, 28, 56 msec. etc., the sample rate can be any value with a resolution of fractions of msec.

Note that the system is designed such that at every time interval, all the information including the teleop is available in the Sun 4/200 RCCL environment. This makes it very easy to implement traded and shared control. By traded control we mean a mechanism by which an operator switches back and forth between autonomous and teleoperated modes. Shared control refers to a 'mix' mode of operation where certain directions are controlled by the operator and the remaining ones by the autonomous system. This applies both for position and force subspaces [6].

2.2 MCM Executive

The remote portion of the MCM system is utilized in two modes of operations: Stand alone and Integrated modes. In the former, one writes C programs in the RCCL environment and executes tasks. This mode is used to develop macros, test and package them so the operator can call them in the supervisory (autonomous) mode of operation from higher levels. In the integrated mode, the MCM executive receives commands from RTC using a communication software called Network Interface Package (NIP) developed at the Stanford Research Institute (SRI). NIP is a layer above DECNET which uses Ethernet hardware to communicate between various subsystems. When a NIP message arrives at the MCM executive, it determines the destination of the command and then directs the data to the appropriate hardware (three puma arms, and two servoed grippers). The MCM executive can receive several commands and queue those that require the same hardware. The higher level systems can always issue cancel commands which will be executed immediately and the queue will be flushed. To make sure that adequate information is provided to the higher subsystems, MCM reports the status of the system approximately once a second.

The NIP software only runs under the VMS operating system. Since the MCM subsystem is Unix based, the NIP commands must first be received by a *gateway* MicroVax running the VMS operating system and then transmitted via DECNET to the SUN 4 computer. The MCM executive can receive its commands either in the form of NIP from the network or from an operator interface provided by the MCM executive. The format of this interactive interface is very similar to MATLAB and is an *extension* to ARC (A Robot Calculator) supported by the new RCCL. ARC is an interactive scalar and matrix calculator which has access to RCCL's numerous robot specific library functions (such as Jacobian, forward kinematics and so on). The extension to ARC attaches it to the actual robots so an analyst can interactively perform certain analysis and then try the results on the robots without compiling or leaving the analysis environment.

3 Multi-Arm RCCL: Multiple Robots and Processors

RCCL (Robot Control C Library) was originally developed at Purdue University [9] and later was improved at McGill University [12]. It was later ported to a MicroVax II [10] and used at JPL, RCA Corporation and McGill University. This implementation still lacked several important features that were needed by the telerobot project's requirements, i.e.,

1. Multi-arm coordination and control capability
2. Adequate computing speed
3. State-of-the-art hardware and software environment for sensor integration
4. Robot independence

The new multi-arm RCCL now implemented has overcome these deficiencies.

3.1 Coordination of Multiple Robots

The original RCCL was designed to program only one arm. Several basic dual arm capabilities are needed to use two or three robots in a convenient manner. These are: independent, synchronized, and coordinated operations. Execution of a single task could easily use all three modes of operations. For example, assume that a pair of robots are utilized to unbolt several bolts from a panel and move it to a different location. If the tool-crib is only reachable by one of the robots, then one robot must pick up a tool and transfer it to the second one. This will require synchronization between the robots. At this point both robots can independently execute unbolting operations. After the bolts are removed the robots must transfer the panel by operating in the coordinated mode.

Coordinated motion requires two Cartesian trajectory generators to realize the trajectories such that certain kinematic constraints are satisfied at all times. To do this, the kinematic ring equation structure was generalized to allow for expressing the kinematic relations for an object which is not a robot [11]. The ring equations for each robot can be kinematically constrained to be attached to this object. When the object moves, so do the robots. The forces of interaction caused by uncertainties in the Denavit Hartenberg [15] parameters and positioning control errors are accommodated by including *COMPLY* transforms which are modified by control laws in functions using force/torque sensor data. Figure 4 shows the planning and control stages of the multi-arm RCCL system.

3.2 Multi-Processor RCCL

Computing power is one of the main limitations both for single and multiple robot control environments. RCCL operates in two levels. The planning level processes the user written code and queues motions for the control level to process. The Control level which provides a real time environment for set point computation is called RCI (Robot Control Interface). The planning level runs in the background (asynchronous) whenever the CPU is available. The control level is that portion of the

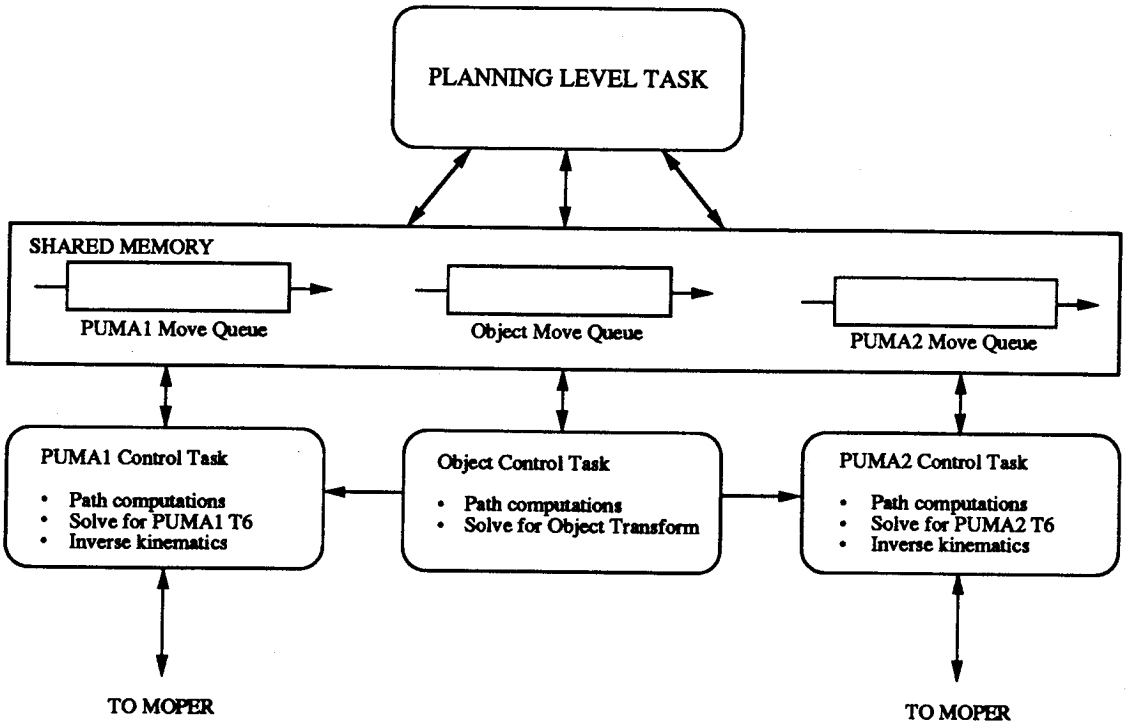

Figure 4: Task Diagram of a Multi-Robot RCCL System

code which determines the sampling rate for the set point generator. The simpler the control level code, the faster the sampling. For example, in the MicroVax II implementation, the sampling rate was 30 Hz. The CPU needed about 14 msec to perform the trajectory generation in the Cartesian mode.

In the new RCCL, the control level software can run either on one or more cpu's. For example, the RCA version uses one MicroVax CPU for the planning level for two robots and two MicroVax II slave cpu's for the two control levels. In the JPL implementation, since the code has been ported to a Sun 4/200 computer, everything runs on a single CPU (note that the Sun 4/200 is approximately 6 times faster that a MicroVax II). Presently, the sampling time for single and dual arm are 200 and 100 Hz, respectively. Note that this is the sampling time for trajectory generation and not for the joint servo control which runs at 1000 Hz. The multiprocessor RCCL makes it very easy to port the control portion of the code to a set of RISC-based boards which at the present time are approximately 2.5 times faster than the Sun 4 SPARC chip.

4 Teleoperation and Shared Control

Teleoperation remains one of the most reliable ways to manipulate objects in a non-assembly type environment. Space operation assembly and repair requires either sophisticated autonomous robotic capabilities or reliable teleoperation. The JPL telerobot approach is to develop both capabilities and advance from pure teleoperation to supervised autonomy and shared control. Effective teleoperation requires telepresence, meaning that the operator feels that he is at the remote site and is operating

the robot rather than the hand controller. Kinesthetic force reflection is an important aid to the operator whenever the robot contacts an environment. Transmission time delay which is unavoidable in space environments decreases the effectiveness of force reflection. The JPL telerobot architecture is designed so that teleoperation experiments with or without time delay can be performed. The adverse effect of time delay can be eliminated by executing tasks in traded or shared control [6].

Teleoperation is implemented by using one UMC and one FRHC for each of the local teleoperation VME chassis. The joint positions are sensed from the UMC's. Cartesian space increments are computed using the hand controller Jacobian and sent to the remote site. This information is received by the MOPER and transmitted to the Sun 4 computer. An RCCL program is written to servo the arm to its present position indefinitely. This null trajectory is then *modified* according to the information received from the hand controller's Cartesian motion. This implementation has two advantages: 1- The trading of control between the autonomous and teleoperation is as simple as executing different MCM *macros*, and 2- shared control can readily be developed. Shared control is accomplished by issuing motions from the autonomous side and modifying them by the motion of the hand controllers. Force feedback to the operator is performed by reading the Lord force/torque sensor *raw stain gauges reading* once every 1.4 msec, transferring the data to the local teleoperation chassis, and applying joint torque computed based on the hand controller Jacobian.

Teleoperation software provides a user friendly menu implemented on the Sun View software. The operator can choose to operate in joint or Cartesian (world and tool) modes of operation and with or without force reflection in the Cartesian mode. For more details see [16]

5 Conclusions

This paper has described the MCM subsystem of the JPL/NASA Telerobot program. MCM is a complete autonomous and teleoperation environment which can be utilized both as a research tool and as a subsystem in the integrated environment of a telerobot. It offers single and dual arm autonomous, teleoperated, and shared control capabilities using state-of-the-art software and hardware for fast real-time computations.

Acknowledgments

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

References

[1] Hayati, S., Wilcox, B., 'Manipulator and Control Mechanization: A Telerobot Subsystem,' Proc. of Workshop on Space Telerobotics, Pasadena, Ca., January 20-22, 1987, pp. 219-227.

[2] Bejczy, A., and Szakaly, Z., 'Universal Computer Control System (UCCS) for Space Telerobots,' Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, Mar. 30 - Apr. 3, 1987, pp. 318-324.

[3] Shanker, P., 'NASA Research and development for Space Telerobotics,' *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 24, No. 5, Sep. 1988, pp. 523-534.

[4] Matijevic, J., Zimmerman, W., Dolinsky, S., 'The NASA/OAST Telerobot Testbed Architecture,' Proceedings of NASA Conference on Space Telerobotics, Pasadena, California, Jan. 31 - Feb. 2, 1989.

[5] Bejczy A. K., Salisbury J. K., 'Controlling Remote Manipulators through Kinesthetic Coupling,' *Computers in Mechanical Engineering*, Vol. 2, No.1, July 1983, pages 48-60.

[6] Hayati, S. and Venkataraman, S., 'Design and Implementation of a Robot Control System with Traded and Shared Control Capability,' Proceedings of the 1989 IEEE Conference on Robotics and Automation, Scottsdale, Arizona, May 14-19, 1989.

[7] Kan, E., et. al., 'The JPL Telerobot Operator Control Station: Part I - Hardware,' Proceedings of NASA Conference on Space Telerobotics, Pasadena, California, Jan. 31 - Feb. 2, 1989.

[8] Kan, E., et. al., 'The JPL Telerobot Operator Control Station: Part II - Software,' Proceedings of NASA Conference on Space Telerobotics, Pasadena, California, Jan. 31 - Feb. 2, 1989.

[9] Hayward V., and Paul, R., 'Robot Manipulator Control Under Unix RCCL,' International Journal of Robotics Research, Vol. 5, No. 4, Winter 1987, pp. 94-111.

[10] Lee, J., Hayati, S., et. al., 'Implementation of RCCL, a Robot Control C Library, on a MicroVax II,' Proceedings of the SPIE conference on Advances in Intelligent Robotics Systems, Vol. 726, Cambridge, MA., Oct. 1986., pp. 26-31.

[11] Lloyd, J., Parker, M., McClain, R., 'Extending the RCCL programming environment to multiple robots and processors,' Proceedings of the 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, April 1988, pp. 465-469.

[12] Lloyd, J., 'Implementation of a Robot Control Development Environment,' Master's Thesis, Department of Electrical Engineering, McGill University, Montreal, Canada, Dec., 1985.

[13] Wilcox, B., et. al., 'Autonomous Sensor-Based Dual-Arm Satellite Grappling,' Proceedings of NASA Conference on Space Telerobotics, Pasadena, California, Jan. 31 - Feb. 2, 1989.

[14] Hayati, S., Tso, K., and Lee, T., 'Generalized Master/Slave Coordination and Control for a Dual Arm Robotic System,' Proceedings of the 2nd International Symposium on Robotics and Manufacturing:Research, Education, and Applications, Albuquerque, New Mexico, Nov. 16-18, 1988, pp. 421-430.

[15] Paul, R., *Robot Manipulator: Mathematics, Programming, and Control*, MIT Press, Cambridge, Mass. 1981.

[16] Lee, T., 'Implementation and Design of a teleoperation System Based on a VME Bus/68020 Pipelined Architecture,' Proceedings of NASA Conference on Space Telerobotics, Pasadena, California, Jan. 31 - Feb. 2, 1989.