A Unified Teleoperated-Autonomous Dual-Arm Robotic System
S. Hayati, T. Lee, K. Tso, P. Backes, and J. Lloyd

This paper describes a complete robot control facility built at the Jet Propulsion Laboratory as part of a NASA telerobotics program to develop a state-of-the-art robot control environment for laboratory based space-like experiments. This system, which is now fully operational, has the following features: separation of the computing facilities into local and remote sites, autonomous motion generation in joint or Cartesian coordinates, dual-arm force reflecting teleoperation with voice interaction between the operator and the robots, shared control between the autonomously generated motions and operator controlled teleoperation, and dual-arm coordinated trajectory generation.

Fig. 1. JPL/NASA telerobot operator control station.
The system has been used to carry out realistic experiments such as the exchange of an orbital replacement unit (ORU), bolt turning, and door opening, using a mixture of autonomous actions and teleoperation, with either a single arm or two cooperating arms.

The goal of this project has been to develop a complete telerobot facility, based on new and existing technology, for performing experiments in the repair and assembly of space-like hardware to gain practical knowledge of such work and to improve the associated technology [2],[3]. In the initial stages, the system can perform a large set of tasks in teleoperation, and a smaller set of tasks autonomously. The JPL telerobot consists of five main subsystems: the manipulator control subsystem (MC), the task planning and reasoning subsystem (TPR), the run time control subsystem (RTC), an operator control station (OCS) [6],[7], and the sensing and perception subsystem (S&P). Fig. 1 shows the operator control station.

The manipulator control subsystem, which is described in this paper, provides a real-time multi-arm programming and control environment with the following attributes: 1) coordinated or independent control of two manipulator arms, and independent control of another arm which carries cameras for direct use by the operator and the sensing and perception subsystem; 2) teleoperation of either arm, or both arms simultaneously, using various modes of operation and with kinesthetic and visual force feedback for the operator; 3) force control capability for contact with the environment; 4) traded control capability so that an operator can switch, between teleoperation and autonomous operations at will; 5) ability to run the system either stand-alone or from a higher level subsystem; 6) shared control capability; and 7) three-dimensional graphical simulation capability for previewing autonomous actions.

This paper is organized as follows. The basic architecture of the manipulator control subsystem is presented, then the multi-arm Robot Control C Library (RCCL), a key software component of the system, is described along with its implementation on a Sun-4 computer. The system’s simulation capability is next described and the teleoperation and shared control features explained.

**Manipulator Control System Architecture**

The manipulator control subsystem controls the three robots and a pair of hand controllers which are used for teleoperation. In terms of the usual architecture of robot systems, it lies below the level usually associated with task and path planning. It is responsible for the real-time generation of trajectories, and for the execution of specific, localized, task macros, e.g., "unscrew bolt," "open panel." The following guidelines were used for the system design.

**Local/remote site separation.** The computational architecture must clearly distinguish between the local site, where the human operator sits, and the remote site, where the manipulators work. Since in space applications the distance between these sites can vary from a few feet to thousands of miles, this decomposition is important because it encourages consideration of the time delay problems that occur when the distance is large. The architecture must use the same remote computational environment to operate the system in either teleoperation, autonomous, or shared control modes.

**Hardware expandibility.** It should be easy to integrate additional hardware (e.g., robots and sensors) into the system.

**Software expandibility.** The system software must be flexible enough to allow the addition or modification of various components, such as a user-defined trajectory generator or a new control law implementation.

**Hardware independence.** Where possible, the software should be kept independent of the hardware (e.g., robots and sensors) through the use of suitable abstractions. It should also be easy to migrate the system to more advanced computing hardware as it becomes available.

**Rapid prototyping.** It should be easy to rapidly develop and test new components and concepts at all levels of the system.

A block diagram of the manipulator control subsystem is shown in Fig. 2. The local site, where the operator is stationed, contains two identical systems (named "left" and "right"). Each consists of a force reflecting hand controller (FRHC) [4] used for teleoperation, a JPL-built Universal Motor Controller (UMC) [1] to provide servo control, a VME chassis with 68020 processors for computing kinematics and force reflections, a Red-Green-Blue (RGB) monitor for displaying force/torque sensor information to the operator, and a Sun-3 computer for the operator interface. The remote site consists of two PUMA 560 arms for which servo level control is provided by two UMCs. The arms are run either independently or in coordination, using an enhanced version of Robot Control C Library (RCCL) [8], [9], [11], [12], which is a library of routines embedded in C for specifying and executing robot motions in either Cartesian or joint coordinates. The enhanced version, called Multi-RCCL, allows the coordinated control of multiple robots and has been implemented on a Sun-4. Another PUMA 560, controlled by a Unimation controller and a MicroVax II system running Multi-RCCL, is used to position a pair of cameras for the S&P subsystem and an additional pair of color cameras which provide the operator with a stereo display of the remote site. This third arm is not controlled using a Sun-4/UMC combination because of the current unavailability of an additional UMC. The two main PUMA arms are each equipped with a Lord wrist force/torque sensor, a served gripper from TRI Inc., and the Unimation teach pendant which operates under RCCL. A dedicated CPU is used to concentrate the

![Fig. 2. Manipulator control subsystem functional block diagram.](image-url)
signals from the local hand controllers, force/torque sensors, servoed grippers, and teachpendsants, and pass the information to the Sun-4 RCCL system; this CPU inhabits the block labeled "MOPER" in the diagram (the name is historical). The complete MOPER unit is a VME chassis containing one concentrator CPU for each robot and several parallel and serial communication cards. The CPUs are Heurikon single board computers, based on the Motorola 68020 microprocessor with a 68881 floating point coprocessor and 1 Mbyte random access memory (RAM) running software developed using VxWorks. Similar equipment is used in the VME chassis attached to the force reflecting hand controllers.

Real Time Operations

Three levels of hierarchy (planning, control, and servo) are present in the control system. The control and servo levels are both "real-time" levels operating at fixed sample rates.

Planning Level (Asynchronous)

The output from this level is an asynchronous stream of individual motion requests for either a single arm or a pair of cooperating arms. Examples of the actions generated by such requests could be "move to position p," or "move in the z direction until a contact force is encountered," or "move to q while complying along the z axis." RCCL provides primitives with which such motions can be specified. It also provides general primitives and data structures for calculations related to robotics, such as manipulating 4x4 homogeneous transforms, or using several such transforms to describe a kinematic loop [13].

A major portion of the system application code is written at the planning level, and consists of C programs which use RCCL to create the motion requests necessary to carry out various tasks. Each motion request is placed on a queue for execution by the RCCL trajectory generator (which runs at the control level described in the next section). Motion execution occurs asynchronously with the planning level; i.e., the RCCL primitives that issue motion requests do not wait for the request to complete before returning. Synchronization can be established explicitly using special primitives.

The planning level executes as a regular UNIX process, and so may interact with other processes, or the outside world, using the conventional UNIX communication facilities. Applications can be constructed to permit direct operator keyboard interaction, or accept commands from higher subsystems via Ethernet and the system Executive. Because it runs as a UNIX process, the ability of the planning level to respond to real-time events is somewhat limited. Applications requiring real-time response have to be written in the control level described next.

Control Level (Synchronous; 100-200 Hz)

This is the trajectory generation level. RCCL provides a trajectory generator which computes the manipulator setpoints required to execute the motions requested by the planning level. The manipulator setpoints are created at a regular sample rate and passed down to the servo level. The trajectory generator performs inverse kinematics (for motions specified in Cartesian coordinates) and provides velocity and acceleration smoothing between path segments. The control level also implements various safety checks, such as those on joint limits and velocities.

Application functions can also be executed at the control level. These are grouped into two categories: tracking functions, which modify the target position of manipulator motions based on appropriate sensor information, and monitor functions, which can do general real-time computations, or use sensor data to check and interrupt particular motions. Monitor functions have been used for various tasks at JPL, such as notifying the teleoperation code when a joint limit or singularity is approached, doing special types of limit checks required in dual-arm control, and establishing force/torque or velocity limits on motions [16]. Tracking functions have been used in conjunction with sensor data read back from the hand controllers and the wrist force sensors to implement teleoperation and compliant force control. These capabilities have been combined to implement shared control [5], [16], which is a mixture of teleoperation, autonomous control, and force control.

The control level is run on the same CPU as the planning task, but not directly under UNIX: to achieve the required regular sample rate, it is connected directly to the interrupt of a device which provides a suitable timing interrupt and run in the context of the device's driver. RCCL allows the control level to be executed on auxiliary CPUs attached to the UNIX host CPU, but implementing this was unnecessary for our purposes. Modifications to UNIX have been made to allow proper exception handling at this level. The sample rate is constrained mainly by the amount of computation the control level requires. If only one arm is being controlled, then a 200 Hz rate is possible, whereas a 100 Hz rate is generally used to control two arms. Communication with the planning level is achieved using shared memory. Communication between the control level and servo level (described next) is done through a separate MOPER module. MOPER is a VME chassis, containing several CPUs and I/O boards, which functions as a data concentrator. It routes sensor and command information flowing to and from the RCCL CPU, the motor controllers (UMCs), the VME chassis controlling the hand controllers, the wrist force/torque sensors, and the teachpendants. This relieves the RCCL computer of the need to do a large amount of interrupt servicing or device polling. MOPER is also a convenient place to implement the hardware specific details of much of the system's robot and sensor interface software.

Servo Level (Synchronous; 1 kHz)

This lower level provides servo control of the manipulator joint positions. Joint setpoints, produced by the RCCL trajectory generator and (possibly) control level tracking functions, are sent to the servo controller via the MOPER module. The position servoing itself is accomplished by the Universal Motor Controller (UMC), which implements a digital PID control loop, plus an auxiliary feedforward signal, and operates at 1000 Hz. The output from the control loop is a PWM signal controlling the voltage applied to the servo motors of the robots. The gains in the PID loop can be changed for particular applications by commands from the higher control levels. If the gains are set to zero, then the auxiliary feedforward signal can be used alone to provide direct voltage control of the motors, with the control loop being closed at the RCCL control level. One can use this to experiment with advanced control laws such as hybrid position and force control, computed torque, and adaptive control (albeit at a lower sample rate). Direct voltage control is used to implement force reflection in the hand controllers.

System Executive

The remote portion of the manipulator control subsystem is utilized in two modes of operations: stand alone and integrated. Stand alone operation corresponds to the user writing C programs in the RCCL environment and executing them. This is mainly used to develop and test task primitives and macros. After a particular task capability is developed, it is incorporated into the larger system where it can be used by the higher level subsystems, which will re-
quest the command either autonomously, or at the request of the operator. This corresponds to the integrated mode. Commands to the manipulator control subsystem are issued by, or through, the run time control subsystem. They are received by a special program, called the system executive, which routes the command to the appropriate manipulator hardware (three puma arms and two servoed grippers) [14]. The interface between the run time control and system executive is an asynchronous queue, similar in style to the queue between the manipulator control subsystem’s planning and control levels. Commands can be placed on the queue, canceled prior to execution, or the entire queue flushed.

**Multi-Arm Multi-Processor RCCL**

RCCL (Robot Control C Library) was originally developed at Purdue University [8] and McGill University [11]. A version was implemented on a MicroVax II system [9] and used at JPL, General Electric, and McGill University. This version still lacked some useful features such as multi-arm coordination and control capability, adequate computing speed, state-of-the-art hardware and software environment for sensor integration, and robot independence. The latest version, Multi-RCCL, has overcome these deficiencies.

**Coordinating Multiple Robots**

The original RCCL was designed to program only one arm. Extending the system to handle several arms entailed generalizing the motion control primitives to allow the specification of different arms. It also required providing the ability to synchronize or coordinate the motions of several arms at once. Synchronization involves organizing the start or stop times of different robot motions with respect to each other. Coordination involves driving the motion target for several arms using a shared tracking function, so that they move in tandem; this is generally necessary when both arms are manipulating a common object and therefore a kinematic constraint exists between them. Execution of a single task can easily require both capabilities. For example, assume that a pair of robots are utilized to unbolt a large 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 is accomplished in Multi-RCCL by specifying a kinematic connection between the target positions of two or more robots, and some common coordinate frame which is sometimes referred to as a "virtual manipulator." Motion requests can be issued to the virtual manipulator, which, when executed, will be followed by the real manipulators whose positions are attached to it: when the virtual manipulator moves, so do the robots. The forces of interaction caused by uncertainties in position servoing are detected using the force information read back from the wrist sensors and resolved using position accommodation techniques.

**Increasing the Processing Power**

Computing power is one of the main limitations in robot control environments, exhibiting itself in the real-time control level. To address this, Multi-RCCL was designed so that the control level software can be run on auxiliary CPUs which share the same backplane as the UNIX host CPU. This feature has been implemented and used to good advantage in a MicroVax II environment, where it has been used to effect cooperative control of two PUMA arms. In the current manipulator control subsystem implementation, the computing power was increased by porting RCCL to a Sun-4 CPU, which resulted in (roughly) a seven-fold speed increase. This is sufficient to provide cooperative control of two PUMA arms, at 100 Hz, without relying on auxiliary CPUs. An auxiliary CPU capability may be introduced into the Sun-4 RCCL at a later date, utilizing auxiliary SPARC boards residing on the Sun-4 VME Bus. This should increase the sample rate associated with the trajectory generation level to about 500 Hz.
and the servo loops could be incorporated into the same hardware. This would significantly enhance the performance of the force control algorithms.

Simulation

RCCL provides a simulation feature under which the control level, instead of connecting to a set of real manipulator arms via MOPER, connects instead to a UNIX program that simulates the actions of the arms. When this is done, the entire RCCL program can be executed as a regular UNIX process, under debugger control if necessary. The basic simulator software bundled with RCCL does only a simple kinematic simulation without modeling of the dynamics or the workspace. This is still useful, however, for developing control level application code.

JPL has also added 3-D graphics to the simulator software, so that one can preview the results of a particular set of motion commands. The graphics support is written using the SUNCORE graphics software. Several options are provided for adjusting the viewing angle, resizing the window, etc. Details may be found in [17].

Teleoperation and Shared Control

Teleoperation remains one of the most reliable ways to manipulate objects in an unstructured environment. Space 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/she 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 when controlling robots from large distances decreases the effectiveness of force reflection. The JPL telerobot architecture is designed so that teleoperation experiments with or without time delay can be performed in the future. The adverse effect of time delay can be eliminated by executing tasks in traded or shared control [5].

Teleoperation is implemented by using one motor controller and one force reflecting hand controller (PHRC) for each of the local teleoperation VME chassis. The joint positions are sensed from the UMCs. 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 serve the arm to its present position indefinitely [15]. 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 autonomous and teleoperation is as simple as executing different primitives, 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 realized by reading the force/torque sensor and transferring the data to the local teleoperator chassis, and applying joint torque computed based on the hand controller Jacobian transpose.

Teleoperation software provides a user-friendly menu implemented on the SunView software (see Fig. 3). The top half of the menu is for command and the bottom half is for system status. Note that the operator can modify the scaling (hand controller motion relative to the robot motion) and the force feedback gains. The operator can choose to operate in joint or Cartesian (world, tool, and camera) modes of operation and with or without force reflection in the Cartesian mode. A commercial voice recognition and synthesizer system is used so that the operator can command the system and receive replies such as “right arm reached singularity” without taking his/her hands off from the hand controllers.

Conclusions

This paper has described the manipulator control subsystem of the JPL Telerobot project. It 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. This robot control environment has been used to develop an interactive system to program and execute single and dual-arm manipulation tasks on the JPL telerobot (see [16]).

References

Paul G. Backes was born in Key West, FL, in 1959. He received the B.S. degree in mechanical engineering from the University of California at Berkeley in 1982, the M.S. in mechanical engineering from Purdue University in 1984, and the Ph.D. in mechanical engineering from Purdue University in 1987. Since 1987 he has been a Member of Technical Staff with the Robotics and Automation Systems Section at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. His current research activities are primarily single and dual-arm sensor-based control including compliance control, force control, shared control, sensor fusion and task execution primitives.

Kam Sing Tso received his B.S. degree in Electronics from the Chinese University of Hong Kong, Hong Kong, in 1979, the M.S. degree in computer science from the University of California, Los Angeles, in 1981, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 1987. Since 1986 he has been a Member of the Technical Staff with the Robotics and Automation Systems Section at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. His current research interests include real-time systems, robotics systems, and fault tolerance.

John Lloyd was born in Victoria, Canada, on April 22, 1958. He received the B.Sc. degree in physics from McGill University in 1980, worked briefly in the wilds of northern British Columbia, and returned to McGill and received an M.Eng. specializing in robotics in 1983. He then worked for a year designing controllers and building laser vision systems for welding robots. During the last two years, Mr. Lloyd has been involved in setting up robot control environments for projects at RCA, General Electric, NASA, and the Jet Propulsion Laboratory. His most recent work has been (with Mike Parker) the development of RCI (Real-Time Control Interface), a package for creating real-time control tasks in a multi-CPU UNIX environment. His research interests include robot kinematics and control, computer languages and real-time computing, and the software engineering aspects of sensor-driven robot and multi-robot systems. He currently lives in Montreal and is pursuing a Ph.D. degree at the McGill University Research Center for Intelligent Machines.