A Prototype System for Ground Control Space Telerobotics

Paul G. Backes, John Beahan, Mark K. Long, Robert D. Steele,
Bruce Bon, and Wayne Zimmerman

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109 USA

Abstract

A telerobotics system is described which provides supervised autonomous control capability for time-delayed ground-remote control applications. The system includes a local site operator interface for interactive task description and a remote site task execution system. Interactive stereo graphics overlay on video is provided at the local site to update the remote environment model. The remote site system is capable of nominal task execution as well as monitoring and reflex motion. Execution utilizes multiple control modules which execute based upon command parameterization.

1. Introduction

Control of space based manipulators from Earth can provide new capabilities for human utilization of space. Ground control of manipulators on manned platforms can be used to perform time consuming, dangerous, or unskilled tasks to allow astronauts to focus on more highly skilled or critical tasks. Ground control of manipulators on unmanned platforms can provide increased flexibility in mission definition and opportunities for new applications, e.g., telescience.

There are various challenges associated with ground control of space manipulators. Communication time delay is a major factor in the design of a ground control system. For example, the round-trip time delay between an Earth based control station and a Space Station based robot control system is expected to be on the order of 8 seconds[1]. The remote space task environment will likely be unmodeled or only partially modeled. The flight component of the ground-remote system will likely have constrained computational power and require flight qualified software. Also, for manned platform applications, such as the Space Station, the space robotic system might be controlled by an operator on Earth or by an operator on the remote space platform [2].

This paper describes the development of a prototype telerobot control system which has been developed for ground control of space manipulators. The system provides supervised autonomy to compensate for the effects of communication time delay. In supervised autonomy, commands are generated through human interaction, but sent for autonomous execution at the remote site. Closed loop control occurs only within the remote site control system. A command can be sent immediately or iteratively saved, simulated, and modified before sending it for execution on the real robot. Early works in supervisory control include [3] and [4]. A more complete description of supervisory control can be found in [5]. Interactive stereo graphics overlay on video is provided at the local site to update the remote environment model. The remote site task execution system provides autonomous execution of commands and command sequences which have been telemetered from the local site. The task execution system also provides monitoring to determine when to transition between commands and reflex action for reacting to anomalous conditions. Task execution utilizes multiple control modules which execute based upon command parameterization.

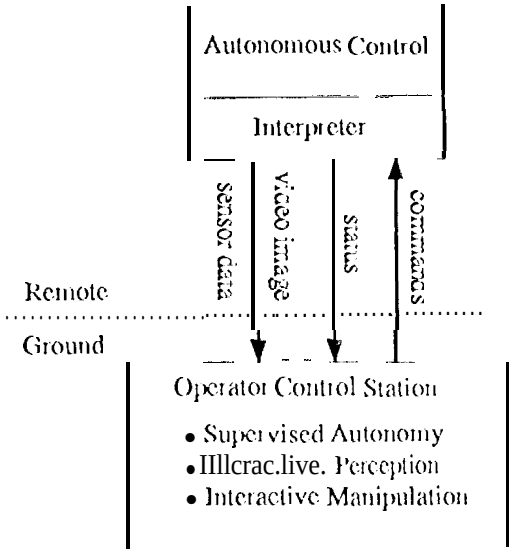

Figure 1: Ground-remote system architecture

The system architecture is shown in Figure 1. The local site generates commands and command sequences which are verified for safe'ty and then sent for execution at the remote site. The remote site autonomously executes the commands and command sequences and sends back periodic status information to the local site. This architecture allows computationally intensive task planning and visualization to reside at the local site where computational power is relatively inexpensive. The remote site does not require autonomous reasoning other than monitoring for conditions specified in the task commands as specified by the local site.

The paper is organized as follows. The local site system is described in Section 2. The remote site system is described in Section 3 and conclusions are given in Section 4.

2. Local Site

The local site' ()])c'rzit[)Ic (]lltlc)lst:\tj c))lllast\vr) primary parts: perception and manipulation. Perception provides an interactive means for modeling the remote site scene. Manipulation provides interactive task description, simulation, editing, and execution. Central to the operator interface is the knowledge base which holds information on the state of the local and remote site systems and Illanipulation and perception data. The methodology of the local-remote system is to build and simulate manipulation and sensing commands at the local (ground) site using a model of the robot and its environment, stored in the knowledge base, which has been updated and validated with feedback sensory data. Simulations detect errors such as kinematic and geometric constraint violations. Successfully simulated commands may be sent to the remote site for execution. The interface between the local and remote sites is data driven interpretive commands for nominal task execution and for reflexactions to monitor events.

The operator control station is hosted on a Silicon Graphics IRIS 310 VGX Power Series workstation, equipped with a 6 axis spaceball input device, and with LCD shuttered glasses for 3-D stereo viewing. An additional X terminal screen can be used, if desired, to host the Graphical User Interface windows, or they may also be hosted on the SGI. The local site software is written in C, utilizing X Windows, Motif, the IRIS Inventor graphics product, and a small library of X resource manager extensions called the Widget Creation Library, developed at JPL. The system provides views from multiple video cameras including a stereo view for depth perception. 3-D graphics is overlayed onto both the stereo video views and the monocular video views in wireframe, transparent, or solid. Also, by displaying only the graphical 111])(1 ['1, the scene can be viewed from arbitrary view']~oillts. Currently, video images are captured a frame at a time. Real-time video capture could be incorporated, but it is not necessary and it might not be available in a flight system.

The interactive perception module provides the ability to update the model of the remote environment. Currently, a graphical geometrical model of the Scene, representing the modeled position of the objects, is overlayed on the video images. The operator can move the geometrical position of the objects in the scene, and simultaneously all children of that object, using the 6 DOF spaceball input device. The operator moves the

graphical objects until they overlay properly on the video images to update the geometric position of the objects. Machine vision could be used to refine the model of the task environment, but his has not yet. been implemented.

The interactive task description and execution part of the graphical user interface is called the User Macro Interface (UMI) and is an evolution of an earlier operator interface [6]. The goal of the interactive task description part of the local site interface is to make task description, verification, and execution as simple as possible to the operator. This is achieved by providing the operator with a library of skills which the remote manipulators can perform. Skills are generic motion types, e.g, Guarded-Motion, Move-To Touch, Hinge, Slide, Screw, Insert, Level, and Push (similar to macros of an earlier system [(i]). When parameterized, a skill becomes a command which can be sent to the remote site for execution of a specific task. The parameterization for a specific skill will depend upon the tool-object pair it is used for. For example, in the command "Insert_ORU_Into_Stowbin" the insert skill is used, but the parameterization is specific to the tool-object pair ORU-Stow bin. Parameterization includes both execution data such as insertion force and impedance parameters and termination conditions such as time, force, or distance. The skill])alalil('tclizatic) llfc)lato ol-ol~jc'et" pair can be prespecified in the interface from design data or input by the operator, and is stored in the knowledge base. After the parameterization for the tool-object pair is entered in the knowledge base, it is automatically available when the tool- object pair and skill are selected. The approach therefore allows the operator to input data into the knowledge base to be used later to simplify future task description and verification. All skills or only selected skills can be available to the operator for a given tool-object pair. To simplify task execution, specific skills can be assigned to specific tool-object pairs. Then when creating a future task, the operator only has to choose from the reduced, appropriate, list of skills. It may eventually be possible for the operator to use the graphical environment to select tool-object pairs and then the appropriate skill could be automatically selected if enough context information is available.

Commands and sequences of commands can be interactively built and saved as a new named command, e.g. "Insert_ORU_Into_Stowbin", and then later recalled for simulation (to ensure that the possibly changed locations of the ORU and stowbin in the knowledge base do not cause errors) and execution at the remote site, Each command in a command sequence includes parameters for various monitors and parameters to specify which monitor conditions are acceptable command termination conditions. If the remote site system terminates a command on an acceptable monitor event, then the next command of the sequence is executed. Otherwise, a reflex action is automatically executed by the remote site system.

The remote site has a fixed software system with task execution behavior dependent on the parameterization from the local site. Multiple control sources can execute concurrently with the resultant motion of the concurrent behaviors providing the task execution. The local site therefore specifies the parameterization for each of the remote site behaviors so that their collective behavior will perform the desired task, The local site also parameterizes reflex actions at the remote site which are executed when monitor events are detected at the rmllotc.site. Examples of individual remote site behaviors and monitors include force control, trajectory generation, joint limit and singularity avoidance, force threshold monitoring, joint limit monitoring, and task space tracking monitoring.

3. Remote Site

Various approaches to programming and control of telerobotic systems have been proposed; each providing a solution for a class of robotic systems. These approaches include environments for programming new applications and languages for general purpose systems. The remote site robot control system described in this paper, the Mod-

ular Telerobot Task Execution System (MOTES) [7], utilizes a command interpreter. The command interpreter is a limited robot language which provides commanding of concurrent control from different control modules. The permutations of control module behaviors are then available to the local site. This method allows a fixed flight software system to provide a wide range of robot control behavior. Also, the command interpreter approach has been proven successful on unmanned robotic spacecraft such as Galileo [8]. The telerobot application is different from the spacecraft application but the system requirements are similar. Various aspects of the MOTES system have functional equivalents to spacecraft control systems. Utilizing the command interpreter approach, MOTES has been designed such that each module is data driven. A command to a module is a parameter set describing the desired behavior for that module. The architecture for trajectory generation and control is fixed but designed to provide a general control capability. Future versions of the command interpreter may encorporate additional language features.

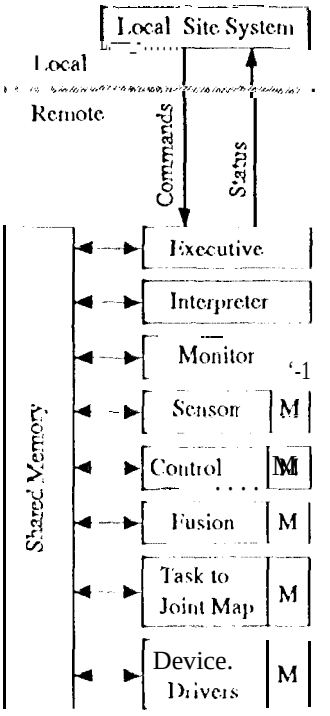

Figure 2: MOTES functional diagram

The MOTES remote site telerobotics task system supports supervised autonomy, shared control, and teleoperation of space robots [7], but only the supervised autonomy capability is used for ground control applications. MOTES is implemented in the JPL Supervisory Telerobotics (STELER) laboratory initially for control of a 7 DOF redundant arm but with planned extensions for dual-arm cooperative control. Capability is maximized by providing simultaneous control based upon various real and virtual sensors. The permutations of the behaviors of the various control modules provides the wide range of capabilities of the system. The desired behavior of each module is specified by commands from the local site which are issued by the remote site command interpreter.

The MOTES system architecture is shown in Figure 2. The functionality of MOTES is similar to the Prim level of the NASREM architecture [9]. This level of a telerobot system generates dynamic motion commands from a static description of the desired behavior. MOTES provides all task level control and task to actuator space mapping. The MOTES module types described below were selected because they represent different functionalities within the control system. There may be multiple modules of the same type, for example force, teleoperation, and collision avoidance control modules.

The **Shared Memory Module** provides access to all command parameters and system status information. The **Executive Module** handles communication with the local site system. It places new commands into the Task Command Queue and returns status and data. The **Interpreter Module** controls the transition between execution states by checking the status of the various modules and specifying the appropriate commands and parameters to the various modules via shared memory. The **Monitor Modules** provide monitoring of the status of execution for both intended termination conditions and unintended

tended error co nditions. The **Sensor Modules** provide sensor data processing. The Sensor modules can represent both real and virtual sensors. The **Control** Modules provide the control associated with the various real and virtual motion sources. Each Control module generates a task space motion command. The **Fusion Module** merges the motion commands of the various Control modules into task space motion commands for the manipulators and other physical devices. The **Task to Joint Map Module** maps the task space command of the Fusion Module to the actuator space of the physical devices. The task space to joint space mapping for the extended task space is done using a composite Jacobian approach [10] using the method described in [11]. The **Device Drivers Modules** communicate with the physical devices to send the actuator space commands and receive status data, as well as perform computations which are hardware specific. It is assumed that the physical devices have their own low level control to implement the actuator space commands. The M in the various boxes of Figure 2 indicates monitoring within the associated 1110(1111(%.

Each module interfaces to the rest of the system through shared memory with specified input and output parameters and functionality. This allows each module to be developed, tested and evolved independently. The modules operate asynchronously with respect to each other with the Interpreter responsible for synchronizing the various modules via modification of command and state parameters in shared memory. Modules 011 a given board may run as fast as possible or be interrupt driven, e.g, clock driven to allow fixed rate computations.

There are various types of commands that the Executive can receive from the local site system. Command types may include Module, Interrupt, Reflex Table, Execution Mode, Initialize, and Emergency Stop. Additional command types, e.g., Cancel, may bc.added in the future.

The MOTES system has been implemented in the JPL Supervisory Telerobotics (STELER) laboratory and controls a 7 DOF redundant manipulator. MOTES is written in the Ada programming language and runs in a VME environment on 68020 processors. The present configuration utilizes six processor boards. Modularity was achieved by developing various modules which could be easily configured onto different Ada tasks running on different boards, e.g., math, shared memory, trajectory generators, form control, teleoperation, impedance equation, forward kinematics, and inverse kinematics. The tasks run asynchronously from each other with some tasks clock driven and others running continuously. Communication between Ada tasks utilizes global shared memory exclusively except for an Ada rendezvous from the Monitor task to the Interpreter task to signal the arrival of a new command. Board memory between tasks and task memory between modules are language supported features but are not used since they would reduce reconfigurability of the system. Module memory is used when appropriate. The global shared memory communication is implemented via Ada generic units. The read and write utilities provided by the generic units provide protection of the data, e.g., complete IWO] '(1 transfers.

In the current implementation, all parameters for one subtask are sent together in one command block. The command type is given as a parameter so that the Executive and Interpreter know how to process the command parameters. Thus data for all modules are placed together in one command and are then parsed out by the Interpreter. The destination queue parameter specifies which command queue to place the command in, e.g., Reflex Command Queues or Task Command Queue.

4. Conclusions

A prototype local-remote telerobot control system has been described. Safety and stability problems associated with communication time delay are accounted for through the use of supervised autonomy where commands and sequences of commands are generated and verified at the lo-

cal site before being telemetered to the remote site for autonomous execution. The remote site task execution system provides a wide range of capabilities with flight qualifiable software by providing multiple simultaneous control behaviors specified through command parameterization.

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.

[1] R. Aster, J.M. de Pitahaya, and G. Deshpande. Analysis of end-to end information system latency for space station freedom. Jet Propulsion Laboratory, Internal Document D-8650, May 1991.

[2] Paul G. Backes. Ground-remote control for space station telerobotics with time delay. In *Proceedings AAS Guidance and Control Conference*, Keystone, C(), February 8-12 1992. AAS paper No. 92-052.

[3] W. R. Ferrell and 'J'. B. Sheridan. Supervisory control of remote manipulation. *IEEE Spectrum*, pages 81- 88, October 1967.

[4] Thurston L. Brooks III. and Thomas B. Sheridan. Superman: A system for supervisory manipulation and the study of human/computer interactions. Technical Report MITSG 7{)-20, Massachusetts Institute of Technology, July 1979.

[5] Thomas Sheridan. *Telerobotics, Automation, and Human Supervisory Control.* M.I.T. Press, 1992.

[6] Paul G. Backes and Kam S. 'J'so. Umi: An interactive supervisory and shared control system for telerobotics. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1096 1101, Cincinnati, Ohio, May 1990.

[-(] Paul G. Backes, Mark K. Long, and Robot D. Steele. The modular telerobot task execution system for space telerobotics. in *Proceedings IEEE International Confe rence on Robotics and Automation*, Atlanta, Georgia, May 1993.

[8] Galileo Project. Galileo program description document - command and data subsystem, phase 9.1. Technical Report 625-3 55-06000, D-535 Rev. G (internal document), Jet Propulsion Laboratory, May 1989.

[9] R. Lumia. Space robotics: Automata in unstructured environments. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1467-1471, 1989.

10] S.Y. Oh, 1). Orin, and M. Bach. An inverse kinematic solution for kinematically redundant robot manipulators. *Journal of Robotic Systems*, 1(3):235- *249, 1984.*

11] Mark K. Long. Task directed inverse kinematics for redundant manipulators. *Journal of Intelligent and Robotic Systems*, 6:241- 261, 1992.