CONF-960448--10
SAND94-3189C

Autonomous, Teleoperated, and Shared Control of Robot Systems

Robert J. Anderson
Sandia National Laboratories
P.O. Box 5800, MS 1176
Albuquerque, NM 87185
bjander@isrc.sandia.gov

Abstract

This paper illustrates how different modes of operation such as bilateral teleoperation, autonomous control, and shared control can be described and implemented using combinations of modules in the SMART robot control architecture. Telerobotics modes are characterized by different "grids" of SMART icons, where each icon represents a portion of run-time code that implements a passive control law. By placing strict requirements on the module's input-output behavior and using scattering theory to develop a passive sampling technique, a flexible, expandable telerobot architecture is achieved. An automatic code generation tool for generating SMART systems is also described.

1: Introduction.

Sandia National Laboratories is developing intelligent robotic systems for unstructured environments. Typical problems include the remediation of large waste storage tanks, the decommisioning of nuclear facilities, and the remote handling of hazardous waste. These tasks must be accomplished in cluttered, poorly modeled environments with limited human access. All of these tasks require a number of mundane repetitive operations such as scanning, scraping and bin disposal, combined with a number of complex teleoperated tasks such as insertion, part acquisition, and camera based path planning. In this paper we describe how SMART: *(Sequential Modular Architecture for Robotics and Teleoperation)* can be used to accommodate all these different modes of operations.

SMART [1] is a modular telerobotic control architecture which combines passive network based modules to implement telerobotic behavior[2,3]. Each module represents a portion of real-time code which implements position/velocity or force perturbations. There are modules for input devices, sensors, kinematics, dynamic filters, constraints, and robots. The operator combines different sets of these modules to implement a behavior. By switching through different behaviors the task is accomplished.

2: Building Autonomous Control Systems.

True autonomous robots do not exist -- at least not at Sandia. All robots receive command and control information from a human operator in one form or another. This interface may be as high level as a CAD-description of a part to be designed and a request to "make part." It may be an intermediate level where the operator teaches points using a teach pendant and then generates a robot program with the robot following the pre-taught points, or it may be a low-level interface, where the operator commands every motion using a direct input device such as a space-ball or force reflecting master.

For our purposes, the term "autonomous operations" will refer to any operation in which the operator interfaces to the robot only at an asynchronous level. Thus, if the operator specifies a series of tagpoints, and tells the robot to move to them one at a time, it is an autonomous operation. If the operator has to continuously press a button on a teach pendant for the robot to move, then it is a teleoperated operation. In this section we will demonstrate how SMART modules can be combined to implement autonomous robot behavior.

2.1: Joint Space Motion.

The simplest type of autonomous robot controller is a joint motion controller, which has two fundamental elements: a trajectory generator and a joint servo-controller. In SMART this is implemented using three modules, a PATH module, a joint servo module (such as the TITAN_JOINTS module), and a termination module (KB1) (Figure 1).

The PATH module generates a joint-interpolated manipulator path for a series of preloaded points. Between each pair of points the manipulator accelerates, moves at constant velocity, and de-accelerates following a trapezoidal velocity profile. By overlapping the acceleration profile of one path segment with the de-acceleration profile of the previous path segment smooth transition speeds are obtained.

A joint servo module such as the TITAN_JOINTS module receives a steady stream of set-points from the previous module and drives the robot accordingly. This can be done by driving the manipulator's actuators directly in SMART or passing the setpoints to a dedicated robot controller.

The termination module, KB1, insures that the velocity commands generated by input devices such as the PATH module, is directed toward the robot. It also provides some additional filtering and reduces wave reflections in the discretized system.

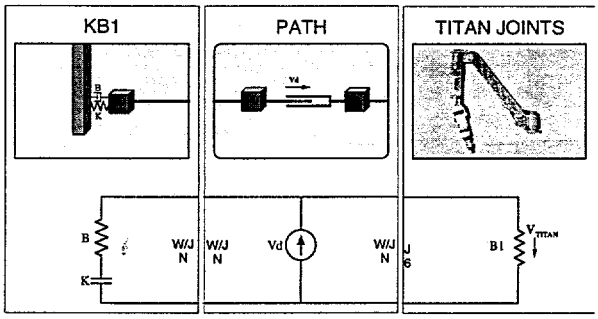

Figure 1: Joint space autonomous controller for Titan

2.2: World Space Motion.

To implement world space motion a kinematics module, such as the TITAN_KIN module needs to be added to the system (Figure 2). The robot kinematics module continually maps the position and orientation of the tool tip to a set of joint angles, and maps the force and velocity between world space and joint space using wave variables.

The trajectory generator operates almost identically in world space, generating straight-line paths between taught positions and orientations. In world space, however, the motion is scaled to the maximum cartesian velocities, rather than the individual joint velocities.

2.3: Graphical Programming.

When working in unstructured hazardous environments most operations are performed infrequently, and cannot be taught using a standard robot teach pendant paradigm. For these operations, researchers have developed a new approach based on interfacing with a 3-D graphical model of the environment using commercially available robot simulators such as IGRIP, SimStation, or RobLine (Figure 3)[4,5] . First, the operator calibrates the graphical world to the real-world using structured lighting, visual targeting, and/or calibration probes. Once the graphical model is deemed suitably accurate, the operator manipulates "tagpoints" in the workcell. The tagpoints store the position and orientation of desired locations in

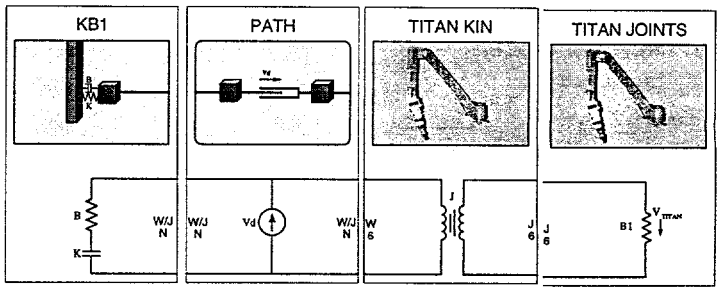

Figure 2: World space controller for Titan

the workcell and also record auxiliary data such as the gripper state, the speed with which to proceed to the next point, or the event that should be triggered when that point is reached.

The supervisory graphical programming system takes the tagpoints depicted in the graphical display and downloads them to the PATH module. Once downloaded it initializes a start motion command. The module will then continuously generate motion profiles until the sequence of points is completed. The connected kinematics module receives the world space positions and orientations and continuously maps them into the local robot joint space, while the joint servo module receives a continuous stream of joint set-points from the kinematics module and drives the robot accordingly. In this fashion autonomous motion of a robot is achieved.

3.0: Teleoperation.

Pure teleoperation requires that the operator directly command all motion of the slave robot in real-time. When the operator is directing motion through the use of teach pendants, space balls or force reflecting masters, the robot should move. When the operator releases the device, the robot should stop.

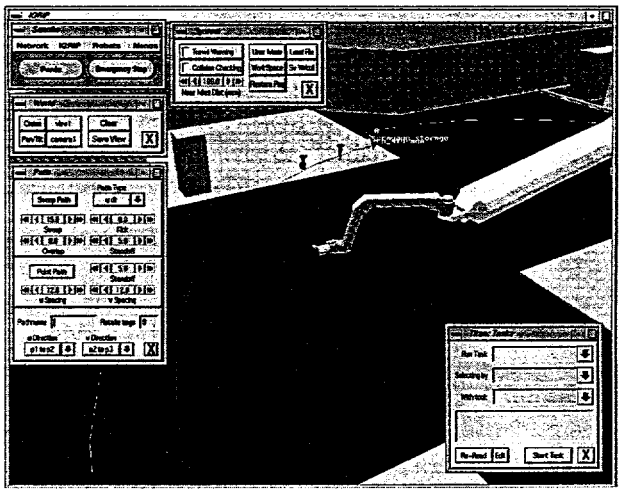

Figure 3:Graphical Programming

3.1 Joint-Based Teleoperation

The most direct teleoperation interface for a robot slave is achieved with a scaled replica master input device. Any motion on any joint of the master is directly conveyed to the similar joint on the slave, and any force detected by a joint on the slave is directly conveyed to the similar joint on the master. If the joint torque signal provides a poor measure of the tool contact forces, then a force sensor may be used in conjunction with the master, but the forces from the sensor must be brought into joint space by mapping through the manipulator Jacobian.

This system may be implemented in SMART using the set of modules shown in figure 4. A force-reflecting master module (in this case the OMEGA_JOINT module which connects to the Schilling Omega force reflecting master) replaces the termination filter on the left-end of the SMART network, and a KBB2 module converts differences in position between the master and slave into a force to drive the slave and to backdrive the master. The JR3_JOINT module maps forces measured from a JR3 force sensor into robot joint space.

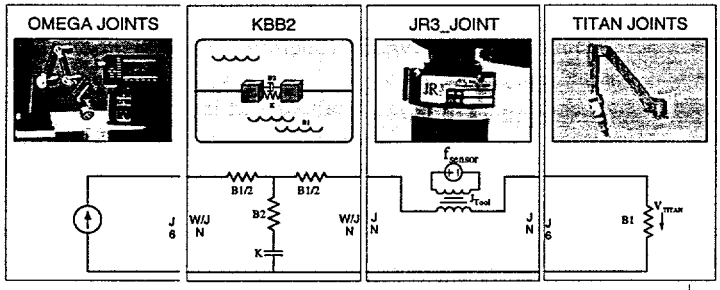

Figure 4: Joint Teleoperation with Force Feedback

3.2: World-Based Teleoperation.

Joint based teleoperation has the advantage of being simple to implement, both algorithmically and computationally. It also has numerous disadvantages. First, a kinematically identical master is needed for every slave to be controlled. Second, it is potentially dangerous since small motions on the master may lead to very large motions on the slave, since master workspaces are typically a fraction of slave workspaces. Third, and most importantly, the resolution of the slave is limited by how accurately the operator can hold the master. For instance, if the master can be positioned precisely only within 1 mm and there is a 1:10 ratio between master and slave, then the slave can only be positioned within 1 cm. This is especially problematic when an industrial robot is used as the slave since the positioning accuracy of the robot is an order of magnitude better than obtainable by a scaled master device. For these reasons generic world based teleoperation has been developed[6].

Generic teleoperation is implemented in SMART by utilizing kinematic modules for both the master and slave, moving the KBB2 module into world space to create a world error function, and adding an INDEX module, which enables arbitrary force and position scaling between master and slave along each degree-of-freedom (DOF) (Figure 5).

3.3: Unilateral Teleoperation.

Bilateral teleoperation, as illustrated in sections 3.1 and 3.2 can provide the operator with important force contact information, but is expensive to implement and is often fatiguing to operators. In many cases, unilateral teleoperation is more appropriate. In unilateral teleoperation, the operator drives the robot with a non-force reflecting input device, either in "rate mode" or in "position mode." In rate mode the operator directly commands the robot velocity, using either a teach pendant or a force/torque ball. Figure 6 shows a SMART system for the Schilling Titan manipulator using the CIS Dimension 6 force/torque ball.

In unilateral teleoperation the operator is dynamically decoupled from the system, and the operator impedance will not affect the system's response. When implemented in the bilateral SMART architecture, however, the force information is still transmitted to the input device module and can still be used to

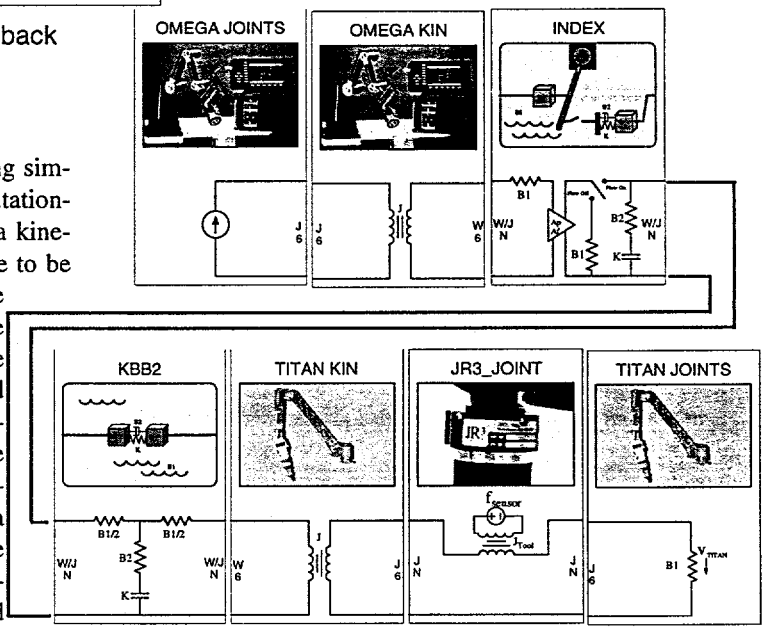

Figure 5: Generic Teleoperation with Force Feedback

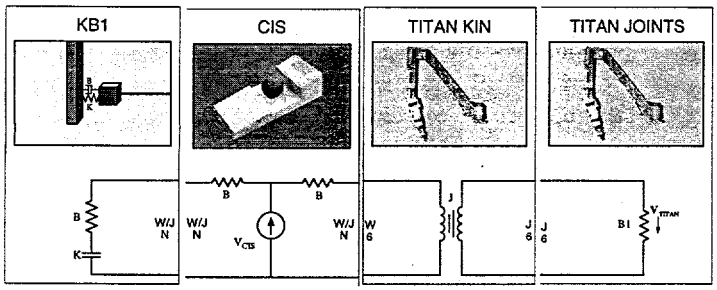

Figure 6: Unilateral Teleoperation

enhance operations. In particular, all unilateral input devices in SMART have a force threshold. If the operator drives the robot so that sensor and virtual constraint modules generate a net force which exceeds this threshold, any further motion in the offending direction is prevented.

4.0 Shared Operations

The previous two sections have shown the extremes of telerobotics, where either the robot is operated solely by a trajectory generator, or solely by the human operator. Generally, a faster, safer solution is obtained by merging human input and autonomous operations, with the appropriate combination depending on the task. In this section a number of methods to achieve this will be discussed.

4.1 Imbedding World Knowledge

One of the first things to add to the teleoperator system is world model information, such as the locations of known objects and the limitations of the slave manipulator. Two SMART modules have been implemented to incorporate world knowledge into the system, the OBSTACLE module and the CLAMP module.

The OBSTACLE module generates a virtual force field around modeled objects in the environment. First the robot tool and the environment are decomposed into convex object primitives. Then every update cycle, the distance between pairs of objects are computed and used to generate nonlinear spring-damper forces along the vectors of nearest proximity[7]. The net force is injected into the system and serves as a barrier force that prevents the human operator from driving the slave into known objects.

The CLAMP module prevents the operator from overdriving either the position or the velocity of the slave robot. A nonlinear damping force is activated whenever any joint's velocity exceeds ninety percent of maximum velocity, and a nonlinear spring-damper force is activated whenever any joint comes within a few degrees of its travel limits. Unlike a linear "coordinating torque" term which tends to inhibit manipulator motion at any velocity, the CLAMP module only slows manipulator motion when actual joint velocities are about to be exceeded. This reduces the fatigue of the human operator, but still ensures the master and slave robots stay synchronized.

4.2: Superimposing Inputs.

One of the most natural methods to share autonomous and telerobotic operations is to superimpose inputs. Here the trajectory generator provides the dominant desired motion, and the human operator perturbs the base motion using a teleoperator input device. For instance, in scanning the waste surface in a waste storage tank a trajectory generator may control the scan path across a surface patch, while the operator controls only the vertical motion of the manipulator. The OBSTACLE module may be used to simulate the contours of the waste surface in this case. Figure 7 shows a superposition system using both the OBSTACLE module and the CLAMP module. Here a one-degree-of-freedom TORQUE_ARM module is used as a force-reflecting master and a MULTIPLEX module is used to determine both the degree of freedom of the coupled motion and the appropriate force and position scaling along that degree of freedom.

4.3: Record, Replay and Training.

Another method to combine human path planning abilities with robot repeatability is to train a system in a virtual world and then replay the motion in the real world.

The REPLAY module uses "motion cassettes" to sample the joint position and virtual manipulator state during the training mode of operation, while the operator drives the virtual system using teleoperator input devices. The VISUAL module provides the communication link to a remote graphics host running a 3-D robot simulator package to provide the operator with a virtual view of the operation.

The motion cassettes are then loaded into the actual system for driving the robot. Since the entire motion path has already been previewed for safe collision-free operation using the graphical simulator, the operator can repeat, slow or reverse motion along the recorded motion path as desired. This is especially useful for operations such as arc welding and cutting in which the operator knows the path a priori, but must carefully monitor the speed along the path during operation based on visual feedback. A simple speed dial can then be used to control the rate of motion along the previously taught path. Figure 8 shows the SMART modules needed to implement both the training and the operational systems.

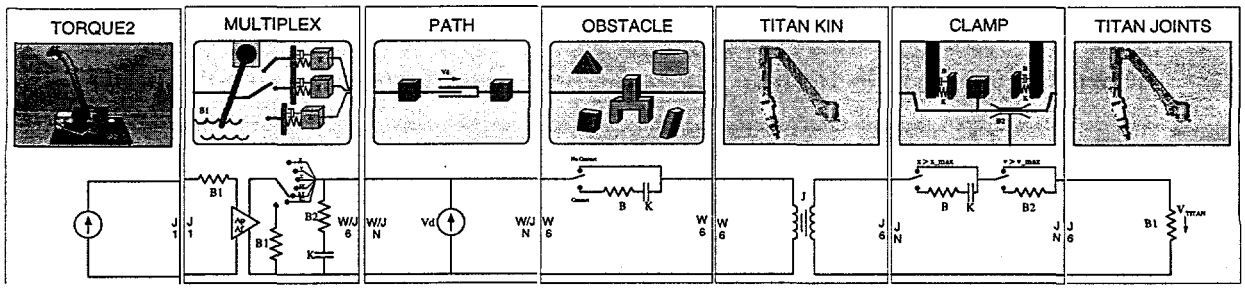

Figure 7: Superposition system with Imbedded World Knowledge.

4.4: Local/Remote Operations.

The SMART system is designed using transmission line principles. These principles have been shown to lead to guaranteed stable behavior for arbitrary time-delay in a delayed master-slave system [8]. Nevertheless performance of a bilateral master-slave system tends to degrade after the time delay exceeds 200 msec. In these cases it is advantageous to dynamically uncouple the teleoperator from the slave system. This can be done by scaling either the forces or the velocities to zero.

Figure 9 shows a bilateral master slave system implemented using two remote SMART networks. The RECEIVE/TRANSMIT module pair allow a SMART system to be arbitrarily connected over ethernet. To dynamically decouple the master and the slave the force scaling on the SCALING module would be set to zero. In this system local virtual forces are still generated on the master side of the system to prevent collision with obstacles and to prevent overdriving the robot. The remote system combines local compliance in the KBB2 module with force feedback from the JR3_JOINTS module to reduce the effects of collisions with the SLAVE manipulator.

4.5 Task Sharing.

In many cases telerobotics is achieved not by having the autonomous system and the human operator sharing every task, but by splitting up the tasks into subtasks which are performed either by the human, by the autonomous subsystem, or by a combination of both. In this case the robot control system must be able to rapidly switch between different modes of operation.

In dismantlement operations for example, the location of manipulator tools are usually pretaught, and the robot will execute an autonomous sequence to grab a tool. The human operator may then teleoperate the robot to places of interest,to drill or cut at a point for instance, and then the system will take over to complete the operation. In surface finishing or scanning the human may teleoperate the robot to first delineate the corners of the region of interest, and then while the robot scans the surface area, the operator controls motion normal to the surface.

5.0 Enabling Technologies.

The different collections of modules shown in the previous sections illustrate some of the behaviors that can be achieved by combining SMART modules. Currently over

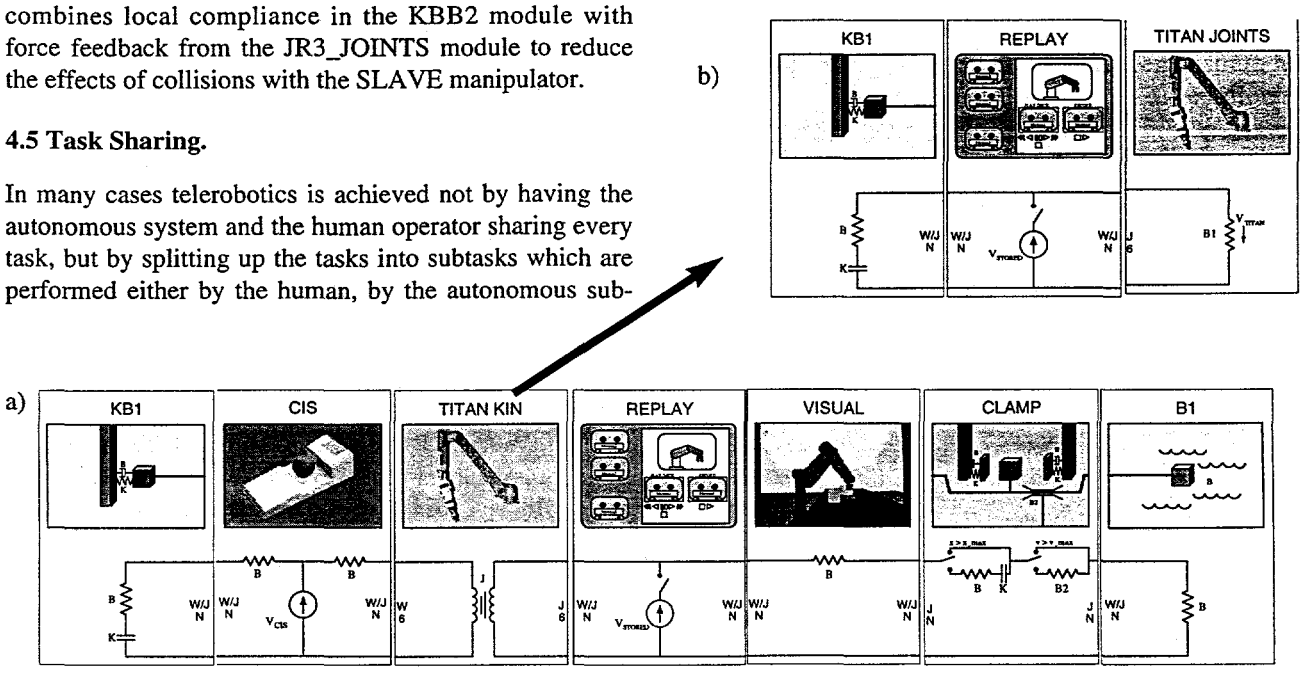

Figure 8: Training system: a) Teleoperator Training in virtual world, b) Operations in real world.

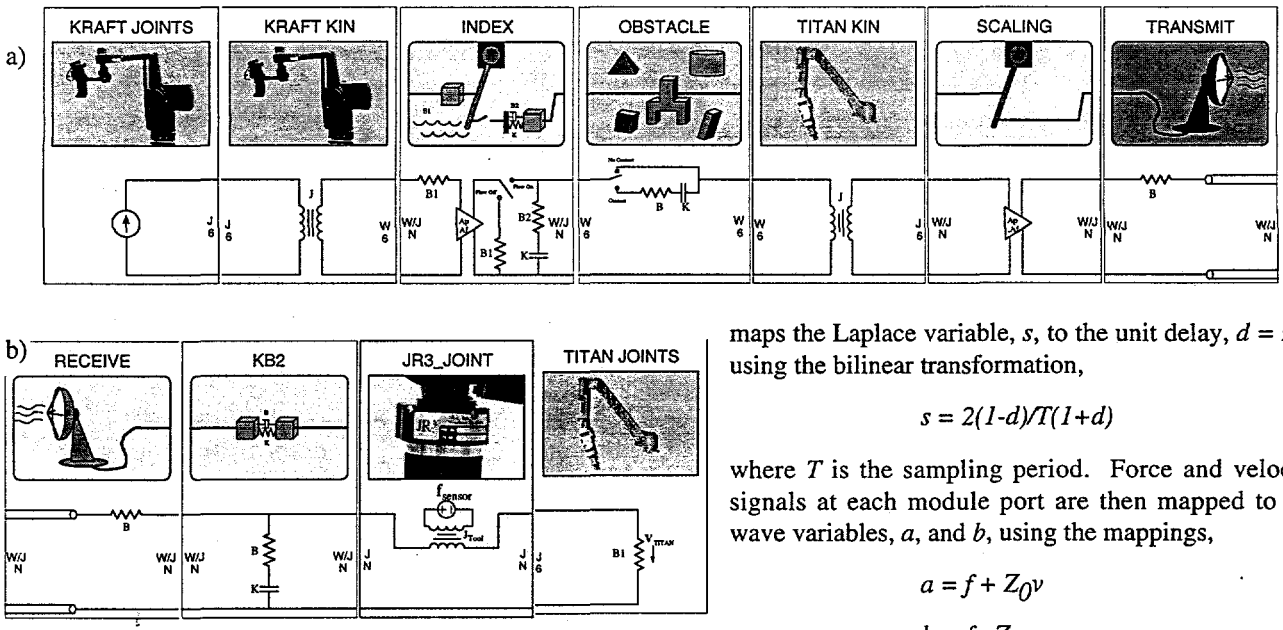

Figure 9: Long Distance Teleoperation: a) Local master system, b) Remote slave system.

a hundred different modules have been coded and tested, including modules for seven different input devices, six different sensors, eight different robots, and various dynamics and constraint behaviors.

A typical telerobotic system in SMART may consist of up to twenty different combinations of these modules each containing from four to thirty modules, which the operator will switch in to achieve different subtasks. A number of technologies have been developed to help the system designer build a system.

5.1 Achieving Modularity in a Discrete System.

The key to the flexibility of building complex modular system is the inherent stability of the SMART architecture. Each module is first designed as a passive subnetwork in the continuous domain. Control laws are implemented in a passive manner by simulating the behavior of passive spring-mass-damper systems connected to independent force and velocity generators. Non-linearities due to kinematic mappings and constraint functions are represented by memoryless Jacobian elements and nonlinear dampers. All dynamic, energy-storing behavior is incorporated into linear-time-invariant sub-networks.

By applying scattering theory techniques each module is discretized while preserving its passivity. For example each dynamic element is mapped to the discrete domain using the passivity preserving Tustin's method, which maps the Laplace variable, s, to the unit delay, $d = z^{-1}$, using the bilinear transformation,

$$s = 2(1-d)/T(1+d)$$

where T is the sampling period. Force and velocity signals at each module port are then mapped to the wave variables, a, and b, using the mappings,

$$a = f + Z_0 v$$

$$b = f - Z_0 v$$

where f is the force across the port on the module, v, is the velocity entering the port. The term, Z0, is called the characteristic impedance. In a transmission line this term would be derived from the line characteristics, and is in general a complex number. In SMART however, this number is a parameter under the designer's discretion. By choosing Z_0 to be a positive real constant, we can guarantee that the output scattering operator mapping for a module,

$$b = S\,a$$

where, S, is the scattering operator, an induced norm less than or equal to one. Connecting the output waves of each module to the input waves of the neighboring modules completes the system.

Since the characteristic of a passive system is that the scattering operator is less than or equal to one, scaling of the wave signal by a sample delay, e^{-sT}, will not affect the passivity of the module, and thus the modules stability behavior is immune to sampling delay. This allows SMART to transfer information between modules in any order, simplifying the implementation of the modular control system considerably.

Furthermore, each module can be tested for nonpassivity simply by driving the module with a series of test input waves at each of the module's ports. If the sum of the squares of the output waves exceeds the sum of the squares of the input waves for a module, then the module is non-passive and cannot be arbitrarily placed in a SMART grid without additional considerations. These

stability techniques are discussed in detail in [9].

5.2: Position Synchronization.

Assuming that each module in the system fulfills the passivity requirements, then stability of the system for arbitrary connections of modules is assured. For a telerobotic system, however, position tracking of the system must also be achieved. A system based solely on force and velocity signals implemented through wave variables would eventually accumulate positioning errors, and the robot motion would deviate from the operator's intended path.

In SMART exact position tracking is accomplished by feeding the position and orientation forward throughout the grid, summing up the net effects of all the input devices, filters, and kinematic mappings until a final set-point position is achieved for the slave robot at the end of the grid. If the force/velocity feedback paths are stable for any position using the scattering operator approach of the previous section, then the modules can use the derived force/velocity information to properly modify the position/orientation feedforward path.

To insure that the position of the input devices is synchronized with the location of the slave robot a synchronization step is initiated every time a new grid is swapped in. During the synchronization stage the feedforward position propagation is reversed, so that the position of the manipulator is propagated back through the kinematics and filter modules until an initial starting location for the teleoperator is derived.

5.3: Implementation Details.

Currently SMART is set-up to run on a VME system under the VxWorks real-time operating system using C-source code. The code is developed on a UNIX host machine and cross-compiled for the VxWorks target central processing units (CPUs) such as the Force 68k, Motorola, and Heurikon processors. If desired, the code may also be cross-compiled for one or more Mercury I860 attached processors. Modules can be arbitrarily distributed across the cpus in the VME backplane. Typical installations will use from two to five CPUs and a single attached processor.

Associated with each module in SMART are a minimum of four required routines, a initialization routine called *module_init()*, a filter constant setting routine called *module_fc_set()*, an update routine called *module_update()* and a print routine called *module_print()*.

Upon initialization the *module_init()* routine is called for each module. This sets up I/O drivers, spawns processes, allocates memory and does whatever else is necessary to initialize a module on a given CPU. It will also set-up the system with default filter constants if none have been pre-assigned by the user. It then spawns the main *smart_system_update()* process, and if on the first cpu, spawns a *smart_monitor* process as well.

The *module_fc_set()* routine is typically called before the *module_init()* routine using the user's predetermined filter constants. It can also be called again at any time to change the current operating mode of the module.

The *module_update()* routine is the main workhorse module. It is called synchronously every few milliseconds by *smart_system_update()* to process the wave and position inputs and to generate the wave and position outputs for the module.

Once the system is running the operator can do a number of things. The operator can change filter constants on any module at any time. He can activate the robot. He can download a set of points to the path module and execute them. He can also swap in a new grid using the *smart_set_grid()* command. This will automatically deactivate any robots, reconnect the modules for the new grid, and synchronize input devices to the robot's position.

The *smart_monitor()* task verifies that all CPUs maintain a heartbeat and continuously updates the state of the system. It ensures that all CPUs are initialized, that all modules are synchronized, and that robot activation commands are acknowledged.

5.4 Automatic Code Generation

To create a SMART system on the host computer, the operator first works with a Graphical User Interface (GUI) called the SMART Editor[Fig 10]. By using the editor the operator defines the different grids of operation by selecting and dragging the SMART icons onto a display field. Once the grids are laid out and verified as valid (i.e., each grid must represent a closed network, and each port must have consistent degrees-of-freedom and orientation states), the operator clicks the "Assign numbers" button. Using internal heuristics and a knowledge of the available hardware, the system will then attempt to assign CPUs and attached processors for each module. In addition to the processor assignment, a configuration file for each module is parsed which contains sets of valid filter constants.

Using a GUI tool the operator can toggle thru the sets of filter constants appropriate for each module by name. Here the term "filter constants" refers to any parameter unique to a module that the operator might want to change

to achieve different operational modes. For instance, the PUMA_KIN module has filter constants describing the Denavit-Hartenburg-(DH) parameters for a PUMA robot. In this case the user selects between pre-determined sets of DH parameters corresponding to PUMA 560 series or PUMA 760 series arms.

Once the user has determined the filter constants and processor numbers for each module in each grid, he clicks on the "Generate Code" button. This will create C-source code for each CPU in the target system, which describes the system, the modules, and the initial filter constants for each module. Numerous heuristics are used based on the chosen set of modules and target hardware platform, to determine an appropriate update period, T, characteristic impedance, Z_0, and distribution of modules accross the cpus. Once generated, the user can include additional application code as necessary.

Finally, pressing the Compile button will result in the code being compiled for each processor, and the appropriate VxWorks startup scripts to be generated. By resetting the target VME the newly developed code will be automatically downloaded and executed.

6.0: Conclusions.

In this paper we have discussed how the SMART system can be used to achieve different telerobotic behaviors ranging from pure teleoperation to pure trajectory tracking. Shared modes of operation included imbedded intelligence, superposition, record and replay, remote and local intelligence, and rapid reconfiguration between any other mode of operation.

To date, SMART has been applied to tasks ranging from force-reflecting bilateral teleoperation, to multi-arm robot coordination, to flexible robot control, to painting with redundant thirty foot robots. The development of the COMM_RECVand COMM_XMIT module pairs combined with special purpose remote viewing technology has allowed us to conduct shared teleoperator development and operations with remote sites.

Current work includes an extension of the OBSTACLE module to include whole arm obstacle avoidance, and the inclusion of more externally generated SMART modules.

7.0: Acknowledgements.

This work was performed at Sandia National Laboratories and supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

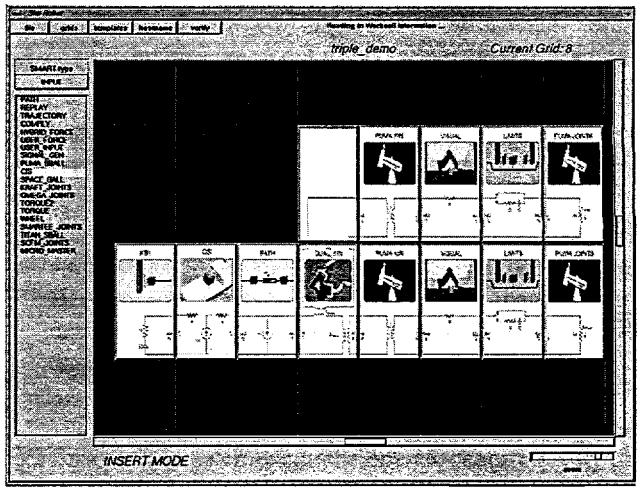

Figure 10: SMART Editor Window

References.

[1] R. J. Anderson, "SMART: A Modular Control Architecture for Telerobotics", *IEEE Robotics and Automation Society Magazine,* Vol. 2, No. 3, Sept. '95 pp. 10-18.

[2] G. Raju, G. C. Verghese and T. B. Sheridan, "Design Issues in 2-port Network Models of Bilateral Remote Manipulation", *IEEE International Conference on Robotics and Automation,* Scottsdale, AZ, pp. 1316-1321, 1989.

[3] R. J. Anderson and M. W. Spong, "Bilateral Control of Teleoperators with Time Delay", *IEEE Transactions on Automatic Control,* vol. 34, pp. 494-501, 1989.

[4] M. McDonald and R. D. Palmquist, "Graphical Programming: On-Line Robot Simulation for Telerobotic Control," *Proceedings of International Robots and Vision Automation Show,* Detroit, Michigan,, pp. 22.59-22.73, April 1993.

[5] J. H. Park and T. B. Sheridan, "Supervisory Teleoperation Control Using Computer Graphics", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, pp. 493-498, April 1991,

[6] A. K. Bejczy, and M. Handlykken, "Generalization of Bilateral Force-Reflecting Control of Manipulators," *Proceedings of 4th RomAnSy, pp 242-255,* 1981.

[7] R. J. Anderson, "Teleoperation with Virtual Force Feedback", *Proceedings of the '93 SPIE Int. Symp. on Optical Tools for Manufacturing and Advanced Automation,* Sept. 1993, Boston, MA.

[8] R. J. Anderson and M. W. Spong, "Asymptotic Stability for Force Reflecting Teleoperators with Time Delay", *The International Journal of Robotics Research,* vol. 11, pp. 135-149, 1992.

[9] R. J. Anderson, "Building a Modular Control System using Passivity and Scattering Theory", *IEEE International Conference on Robotics and Automation,* Minneapolis, Minn., April 22-28, (accepted for publication), 1996.

DISCLAIMER