Janez Funda
Thomas S. Lindsay
and
Richard P. Paul
GRASP Laboratory
University of Pennsylvania,
Philadelphia, Pennsylvania 19104

Teleprogramming: Toward Delay-Invariant Remote Manipulation

Abstract

This paper addresses the problem of teleoperation in the presence of communication delays. Delays occur with earth-based teleoperation in space and with surface-based teleoperation undersea using untethered submersibles and acoustic communication links. The delay in obtaining position and force feedback from the remote slave arms makes direct teleoperation infeasible.

We are proposing a control methodology, called *teleprogramming,* which draws on the experience in the development of supervisory control techniques and robotics over the last three decades and introduces a number of new ideas in operator–model interaction as well as the nature and content of the information being sent to the slave robot. A teleprogramming system allows the operator to kinesthetically, as well as visually, interact with a graphic simulation of the remote environment and to interactively, on-line teleprogram the remote manipulator through a sequence of elementary robot instructions. A key feature and contribution of this work is the fact that these instructions are generated automatically, in real time, based on the operator's interaction with the simulated environment. The slave robot executes these commands delayed in time and, should an error occur, allows the operator to specify the necessary corrective actions and continue with the task.

We will in this paper introduce the overall teleprogramming control concept, describe its main components, and report on the preliminary results using our experimental teleprogramming system.

1 Introduction

There is a clearly established and growing need to perform work remotely. Typical applications are tasks such as deployment, maintenance, and retrieval of satellites in low earth orbit, maintenance and repair of underwater man-made structures (communication cables, oil drilling installations), and acquisition of marine and geological samples in underwater environments. These and other tasks require the ability to control a robotic system remotely. While an autonomous robotic capability would solve the problem, such capacity remains beyond the state of the art in robotics (Stark, Kim, & Tendick, 1988). Autonomous systems require sophisticated environment modelers and on-board reasoning systems to adapt to unexpected circumstances. To date, expert systems and other rule-based systems, restricted to reasoning within the domain of a limited and discretized knowledge of the world and a predetermined set of inference rules, have failed to adequately model and adapt to the complexities of the real world environments, and in particular, robotic interactions in unstructured environments.

Presence, *Volume 1, Number 1, Winter 1992*

Consequently, teleoperation remains the most reliable and practical approach to performing work remotely. However, large physical separation and/or low-bandwidth communication media introduce transmission delays between a remote workcell and a ground-based station, rendering direct teleoperation infeasible. A teleoperated human–master–slave system is a closed-loop feedback system and as such very sensitive to delays in the control loop. In particular, as human neuromuscular control of the limbs and fingers operates at approximately 5 Hz (Brooks, 1990), feedback delays in excess of one-third of a second become troublesome to the operator, whereas delays in excess of 1 sec render direct teleoperation impractical for most applications (Ferrell, 1966; Black, 1971). In contrast, roundtrip communication latency between an earth-based control station and a robotic workcell in low earth orbit ranges from 2 to 8 sec (Bejczy & Kim, 1990). Similarly, acoustic transmission between a surface support ship and an untethered submersible at a depth of 1 mile introduces communication delays in excess of 2 sec (Sheridan, 1990).

The problems associated with feedback delays in teleoperation were recognized as early as 1965 (Ferrell, 1965). Early approaches to overcoming the adverse effects of delays included slowing down the motion, introduction of passive compliance to avoid damage to the robot and the environment on impact, and adoption of the "move-and-wait" control strategy (Ferrell, 1965). However, it soon became apparent that greater autonomy needed to be delegated to the remote workcell to allow for improved efficiency of task performance and stability of the overall human–master–slave control loop. With the advent of faster and more powerful computers, the possibility of automating low-level control functions at both the operator's control station and at the remote site was recognized. Such a control paradigm would relieve the operator of involvement in the low-level execution-monitoring and decision-making process and place the operator in a supervisory role of high-level goal specification and error intervention. This approach was formalized as the supervisory control concept (Ferrell & Sheridan, 1967) and became the dominant model for research in time-delayed remote manipulation. Among the recent approaches to alleviating the adverse effects of feedback delays in telemanipulation are formal modeling of up-link and down-link delays as part of the dynamic model of the remote environment (Hirzinger, Heindle, & Landzettel, 1989), devising delay-tolerant control laws based on formal two-port network models of teleoperated systems (Raju, 1989; Anderson & Spong, 1988; Hannaford, 1989; Yokokohji & Yoshikawa, 1990), demonstrating manipulation strategies where control functions are shared between the human operator and the remote slave (Bejczy & Kim, 1990), and design of sophisticated predictive graphic displays (Noyes & Sheridan, 1984; Bejczy, Kim, & Vencura, 1990).

Full realization of supervisory control methodology in remote manipulation has been hampered by the difficulty of adequately automating the low-level physical interaction between the robot and the environment at the remote site. Another problem is the need to anticipate, detect, and provide preprogrammed corrective actions for the multitude of possible error conditions arising during subtask execution to support the necessary level of autonomy at the remote site. With error handlers themselves being subject to errors, the error handling code can easily come to dominate an application program as well as the programming effort itself.

In view of these obstacles, this paper introduces a teleprogramming control methodology, whose central feature is a new approach to low-level task specification for a remotely located robotic system and which offers a practical intermediate solution to time-delayed remote manipulation. Teleprogramming combines the power of a graphic previewing display with the provision of real-time kinesthetic[1] feedback, to allow the operator to interactively define the task to be performed remotely. The locally closed, high-bandwidth feedback loop at the operator's station allows for stable interaction between the operator and the simulated task environment, and the immediate visual and kinesthetic feedback provide for a strong sense of model-based teleperception.[2] Moreover, the system continuously monitors the operator's actions in the simulated environment and extracts from this ac-

1. *Kinesthesia*—sensory experience derived from muscles, tendons, and joints and stimulated by bodily movements and tensions.

2. *Teleperception*—indirect awareness (via the teleoperator) of the remote environment through physical sensation.

tivity a stream of elementary robot instructions, capturing the essential features of the task in progress. The action interpretation and commands stream generation process is guided by a priori information about the nature and goals of the task. The resulting instructions are symbolic in nature and at the level of guarded and compliant motion primitives to allow for discrepancies between the world and the operator's station-based model. They are generated automatically, on-line as the task progresses and are sent to the remote workcell incrementally, as they become available. The remote site receives these instructions a transmission delay τ later, translates them into the local control language, and executes them (delayed in time) under the control of a local high-bandwidth sensory feedback controller. Due to modeling, sensing, and control errors, execution failures may occur at the remote environment. On detecting an error, the slave sends all relevant information about the error state to the operator's station. There this information is used to alert the operator to the error condition, properly adjust and update the graphic world model, and allow the operator to specify the necessary corrective actions and proceed with the task. Figure 1 illustrates the high-level conceptual design of the teleprogramming control paradigm.

In the following sections we will first present the motivation behind the development of a teleprogramming system. We will then describe the teleprogramming concept in more detail within the framework of our current experimental teleprogramming system. Finally, we will present the results of some preliminary experiments using our prototype laboratory system to accomplish some simple tasks and conclude with some comments on future directions and the potential applications of this technology.

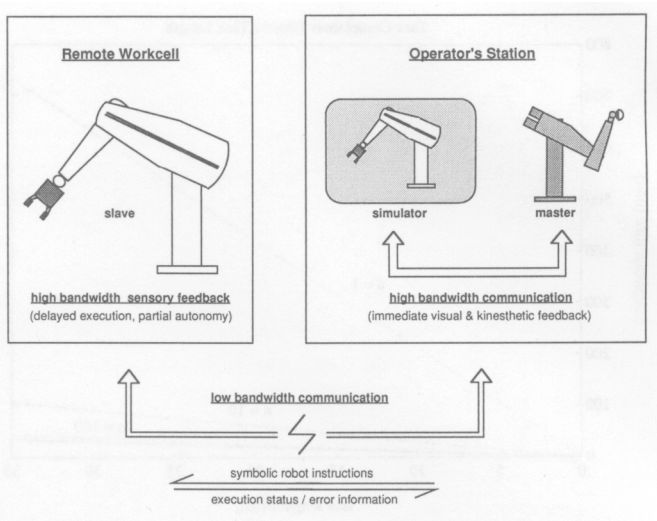

Figure 1. *A high-level view of teleprogramming.*

2 Motivation

As described above, direct teleoperation degrades to move-and-wait control in the presence of significant feedback delays. In particular, if we let the total time to perform a task in a delay-free environment be T_{task} sec, and denote the average elementary motion duration by t (sec), and the total sum of all one-way delays in the control loop by τ (sec). Then, a simple model of the move-and-wait control strategy gives the total time to accomplish the task as

$$T_{total} = \left(1 + 2\frac{\tau}{t}\right) T_{task} \tag{1}$$

In contrast, assume that the remote robotic system is capable of sufficient autonomy to allow the operator to perform, on the average, n instructions of length t without an error occurring at the remote site. Then, in view of Eq. (1), the total completion time becomes

$$T_{total} = \left(1 + 2\frac{1}{n}\frac{\tau}{t}\right) T_{task} \tag{2}$$

Clearly, in the interest of minimizing the overall completion time, the desired behavior of such a system is

$$nt \gg \tau \tag{3}$$

Figure 2 illustrates total completion times (T_{total}) versus task length (T_{task}) for $\tau = 10$ sec, $t = 1$ sec, and three different values for n. Note that the case of $n = 1$ corresponds to the move-and-wait strategy [Eq. (1)] and the solid line on the very bottom corresponds to direct teleoperation with no delay ($T_{total} = T_{task}$). The figure suggests that even a relatively modest amount of remote site autonomy, for example, $nt = \tau$, dramatically improves

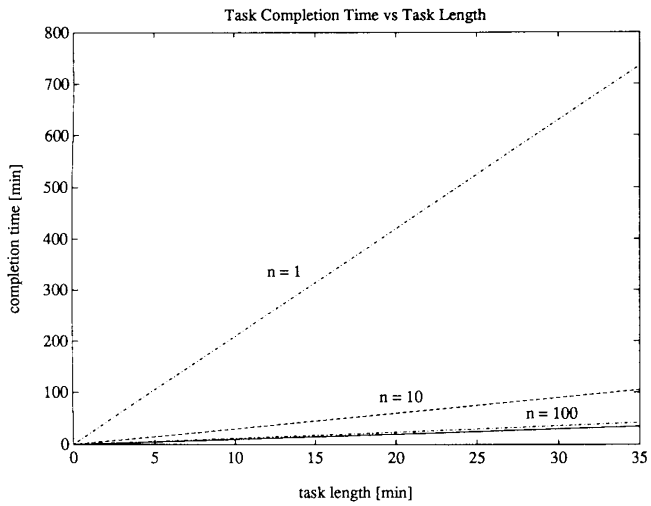

Figure 2. *Total task completion times versus task length for* $\tau = 10$ *sec,* $t = 1$ *sec, and three different values of* n.

the system's throughput (task completion times), whereas autonomy at the level of $nt = 10\tau$ results in completion times that are only slightly longer than the times obtained with direct teleoperation with no delay in the loop at all. For low earth orbit and underwater applications we normally have $\tau < 10$ sec and so the maximum required amount of autonomy for efficient telemanipulation in these domains is on the order of 100 secs.

It is the achievement of this sustained level of remote site autonomy that has hampered the realization of reliable and efficient supervisory telerobotic systems. The teleprogramming control paradigm aims at achieving the necessary level of autonomy and reliability at the remote site by communicating the task to the remote robot in terms of robot instructions. The operator's station thus incrementally converts a task specified by the operator into a robot program. The instructions contain estimates of the modeling errors (between the graphic model and the actual environment) and consist of guarded and compliant contact motion primitives, which are executable with a high degree of reliability under local sensory supervision at the remote site. Therefore, if the task being specified by the operator in the modeled environment can indeed be automatically converted into a stream of such instructions, then, using state-of-the-art modern robot control strategies, remote site autonomy at the level of 100 sec should be achievable in most cases. This suggests that the teleprogramming control paradigm can be successfully applied in low earth orbit and underwater environments, effectively eliminating the adverse effects of transmission delays, and allowing for near-optimal remote control of robotic workcells in these environments.

3 The Teleprogramming System

3.1 Modeling the Environment

We propose that the initial description of the remote environment be obtained by the remote workcell on arrival to the designated work area through the use of on-board sensors, such as vision, touch, or dense range data. This raw data will be sent to the operator's station, where the operator will interact with the image processing module, providing high-level interactive guidance to the data fusion and segmentation stages of the scene reconstruction process. The segmented representation of the scene can then be approximated with polyhedral objects, resulting in a three-dimensional polyhedral representation of the remote environment in terms of objects, environment features, and equipment parts. Moreover, calibration and specification information, pertaining to the sensors that supplied the original data, together with the accuracy of the polyhedral approximation, can be used to define upper bounds on the positional uncertainty of objects in the reconstructed environment. The necessary component technologies for such a system are within the state of the art in computer vision (Srihari, 1981; Besl & Jain, 1985; Bolle & Cooper, 1986).

We currently assume that a polyhedral model of the remote environment is available and that the upper bounds on positional uncertainties in the model are known. The model is displayed on a graphic workstation, providing a realistic three-dimensional view of both the remote robotic system and the surrounding environment.

3.2 Controlling the Motion of the Slave

To specify motion parameters to the simulated slave robot (and, indirectly, to the remote workcell) a 6

d.o.f. input device needs to be interfaced to the graphic model. In our system the master input device is a 6 d.o.f. general purpose robot manipulator with a 6 axis force/torque sensor mounted at its tip. Through the handle, the operator exerts forces on the sensor, which in turn, via filtering and scaling, provides incremental Cartesian displacements to the master arm. The master arm is actively servoed (in position mode) during operation. This arrangement allows for programmable impedance at the master and affords sufficient workspace volume to give the operator a true sense of six-dimensional spatial maneuvering. The mobility of the master is also essential to support the kinesthetic pseudo-force reflection feature as described in Section 3.5.

The motion of the master is coupled to the motion of the simulated slave by establishing a Cartesian correspondence of motion between the terminal links of the master device and the slave robot. The system allows for arbitrary scaling of the relative motions between the master and the slave, and manual as well as automatic reindexing of the master arm. The latter is provided to avoid kinematic singularities and workspace constraints on the master device.

3.3 The Graphic Simulator

Because we are presumably operating in unstructured and largely unknown surroundings, many of the dynamic parameters of the remote environment, such as masses, inertial parameters, and frictional properties, will not be known a priori. This, along with the difficulty of adequately modeling effects such as hydrodynamics and buoyancy in an underwater application, suggests the use of a nondynamic, kinematic simulation of the remote environment, including the slave robot.

The primary task of the graphic simulator in a teleprogramming system is to provide a real-time graphic animation of the slave motions commanded by the operator via the master arm. The graphic display should provide a realistic view of the remote environment, the slave arm(s), and afford the operator control over the viewing direction, zoom, and other critical perspective parameters.

Second, the simulator software continuously monitors the slave robot and any object in its grasp for collisions or contacts with the environment. To reduce the computational burden on the system and to allow the system to distinguish between desired and undesired collisions, the system should be given an indication of which pairs of objects in the environment are expected to come into contact during the execution of the task. This information is part of the task model and will be addressed in more detail in Section 3.6. Due to the potentially large number of objects that may need to be monitored for interpenetration, there is a need for an efficient polyhedral collision detection algorithm. Our experimental teleprogramming system (Section 3.10) makes use of a fast, near-linear distance estimation algorithm for convex polyhedra (Gilbert, Johnson, & Keerthi, 1987), which is used as the basis for computing the set of currently active contacts between the simulated slave and its environment. In particular, if $\mathbf{x}_A$ and $\mathbf{x}_B$ denote the closest points[3] on a pair of convex objects A and B, then the *distance variation* $\Delta d = \Delta|\mathbf{x}_B - \mathbf{x}_A|$ due to an incremental displacement $(\Delta\mathbf{t}, \Delta\mathbf{r})$ of the slave's end-effector can be expressed as

$$\Delta d = \mathbf{n}(\Delta\mathbf{x}_B - \Delta\mathbf{x}_A); \qquad \mathbf{n} = \frac{\mathbf{x}_B - \mathbf{x}_A}{|\mathbf{x}_B - \mathbf{x}_A|} \tag{4}$$

where $\mathbf{n} = (\mathbf{x}_B - \mathbf{x}_A)/d$ and $\Delta\mathbf{x}_A$ and $\Delta\mathbf{x}_B$ are the positional displacements of the points $\mathbf{x}_A$ and $\mathbf{x}_B$, respectively, due to the slave arm displacement $(\Delta\mathbf{t}, \Delta\mathbf{r})$. We define the objects A and B to be *in contact* whenever $d < \epsilon$, where ϵ is a small positive distance that is imperceptible to the human eye, but keeps the mathematics of collision computation well behaved.

Clearly, a positive Δd indicates that the motion causes the objects to be separated further apart. Even if Δd is negative, there is no danger of collision as long as $|\Delta d| < (d - \epsilon)$. Otherwise, the intended incremental motion $(\Delta\mathbf{t}, \Delta\mathbf{r})$ will cause a collision and must thus be modified to apply only the allowable portion of the motion, that is, to stop the offending motion in a nonpenetrating

3. In this context *closest* is defined in terms of the minimum Euclidean distance.

contact configuration. The allowed fraction of the motion is given by the *contact coefficient*

$$t = \frac{-(d - \epsilon)}{\Delta d} \tag{5}$$

For each contact the system extracts the relevant information capturing the nature of the geometric constraint that it imposes on the motion of the slave robot. In particular, for each contact the system records the quintuple

$$c_i = \{f_1, f_2, {}^B\mathbf{p}, {}^B\mathbf{n}, {}^B\mathbf{e}\} \tag{6}$$

where $f_i \in \{vertex, edge, face\}$ represent the contact features of the robot, a tool, or a grasped object (hereafter referred to as the *moving object* or MO) and the environment, respectively. The vectors **p, n,** and **e** denote the contact point location, constraint normal, and edge direction (in case of a line contact), respectively. The set of all (N) currently active constraints on the motion of the simulated slave is thus stored as the *contact set C,* where

$$C = \bigcup_{i=1}^{N} c_i \tag{7}$$

This information is then used both to restrict subsequent commanded motions to the simulated slave so as to not violate any of the environmental constraints (Section 3.4), and to provide real-time kinesthetic feedback to the operator by restricting the motion of the master input device (Section 3.5).

3.4 Motion Restriction

3.4.1 Classification of Motion Modes. A teleoperation system must provide a wide range of motions both in free space (while approaching/leaving the work area) and in contact with the surroundings (while performing the work). At the same time the system should strive to minimize the complexity of slave/environment interaction to increase the reliability of execution of the corresponding actions in the remote environment, as well as allow the operator to restrict attention to only those motion parameters that may be relevant to the current task goal. In view of this, our teleprogramming system offers three free-space motion modes (**free, translate,** and **rotate**), and three contact motion modes (**freeze, slide,**

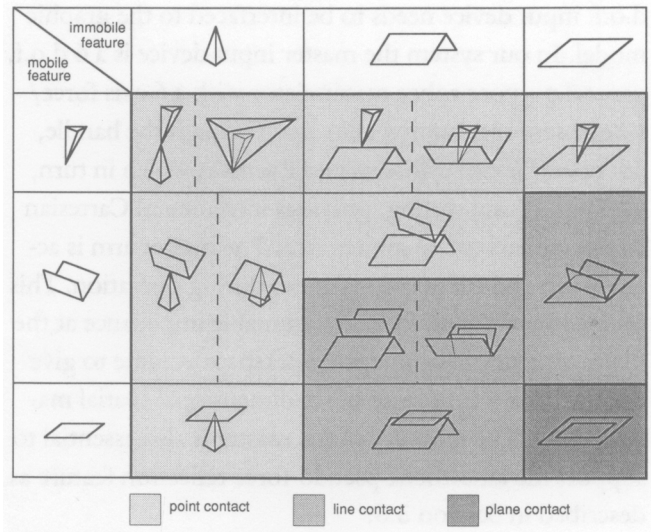

Figure 3. *Types of polyhedral contacts.*

and **pivot**), with the obvious semantics. To minimize the burden on the operator, mode switching is done automatically, based on operator's kinesthetic input. In cases where this is not possible (e.g., selecting pure translation during free space motion), other information channels, such as voice or handle-mounted pushbuttons, can be used. Also, audio confirmation of the current mode is provided to ensure consistency with the operator's expectations.

3.4.2 Contact Motion Computation. When the moving object (MO) is in contact with the (simulated) environment, its motion (and therefore the motion of the slave manipulator) is restricted, depending on the type of contact. Following the notational convention of Sawada, Ishikawa, Kawase, and Takata (1989), Figure 3 lists the types of contacts that we consider in this work (we are concerned with rigid polyhedral contacts only). In view of Figure 3 we will refer to an *adjacent contact* as one that can be reached in one contact change from the current state, and we will say that a contact c_i is *higher* (of higher order) than contact c_j, if c_i offers fewer remaining d.o.f. of motion than c_j.

As described in Section 3.3, the graphic simulator monitors the motion of the slave manipulator and maintains the set of all current contacts between the MO and the environment in C. Depending on the number and

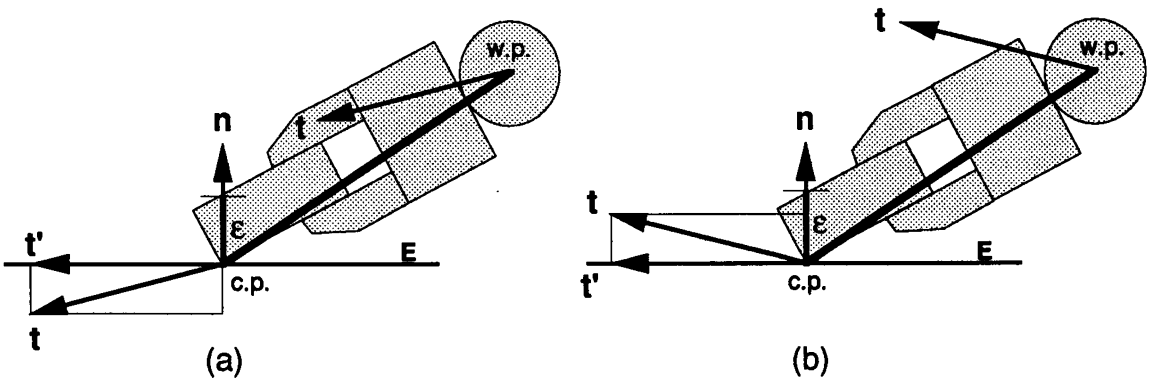

Figure 4. *Single contact sliding.*

types of these contacts, the motion of the slave robot is restricted along one or more d.o.f.. To determine the restrictions, a *contact frame* $\mathcal{F}_C$, aligned with the contacting object features and centered at the *primary*[4] contact point (c.p.), is constructed. The contact set C is then projected onto the contact frame $\mathcal{F}_C$ and the *constraint set* S is computed as the set of orthogonal constraints on the motion of the MO. We use this constraint set in two ways. The first is described below, and the second is addressed in Section 3.5.

Within the framework of the graphical simulator, the constraint set S is used to *restrict* subsequent commanded motions to the simulated slave manipulator so as to not violate any of the geometric motion constraints. This allows the simulated slave to stop on contact with environmental objects, slide along surfaces, follow edges, pivot from a *vertex/face* into a *face/face* contact, etc. The restriction operators for sliding and pivoting contact motions are described below.

Sliding mode. In this mode, the operator can slide the MO along the constraining feature(s) (surfaces, edges) in the unrestricted directions (i.e., the directions not violating *any* of the current constraints). The orientation of the MO remains fixed for the duration of motion in this mode.

As an example, Eq. (9) and Figure 4 illustrate the case of single contact sliding, where an object (grasped firmly by the slave) is slid along an environment surface. Given the desired motion of the slave wrist (w.p.) $\Delta\mathbf{d} = (\mathbf{t},\mathbf{r})$, we compute the corresponding allowable subset of the translational motion $\Delta\mathbf{d}'$ as

$$\Delta\mathbf{d}' = (\mathbf{t}', \mathbf{0}) \tag{8}$$

where

$$\mathbf{t}' = \begin{cases} \mathbf{t} - (\mathbf{t}\cdot\mathbf{n})\mathbf{n}, & \text{if } (\mathbf{t}\cdot\mathbf{n}) < \epsilon \\ \mathbf{t}, & \text{otherwise} \end{cases} \tag{9}$$

Note that the commanded motion is modified (restricted) by removing the components that would cause surface penetration. Note also that in case (b) of Figure 4 the commanded translational motion $\mathbf{t}$ produces sliding motion, rather than breakage of contact. This is due to the "surface adhesion" coefficient ϵ, which is intended to aid the operator preserve achieved contacts.

Pivoting mode. Alternatively, while in contact, the operator can adjust the orientation of the MO or transition between adjacent contacts by rotating or pivoting about the (primary) contact point. As the contact type (between the MO and the environment) changes, the contact point moves on the surface of the MO and with it the pivoting point about which rotational motions are computed. This allows a variety of reorienting and contact changing motions of the MO. Again, motion analysis is performed on the commanded displacements so as to aid the operator to perform the desired changes of orientation. The system provides a restricted version of this motion modality also in situations where multiple constraints are restricting the motion of the MO.

4. In case of multiple contacts, the *primary* contact point is taken as the one absorbing the largest component of the operator's exerted effort.

The restricted contact rotation $\Delta\mathbf{d}' = (\mathbf{0},\mathbf{r}')$ for the three types of single contact pivoting is computed as illustrated in Figure 5. As with sliding, a small amount of contact bias is built into the system to aid the operator in preserving (the presumably desired) higher order contacts. For a given pivoting surface adhesion coefficient ξ, the restriction operator Υ is given as

$$\Upsilon(x) = \begin{cases} 0, & \text{if } |x| < \xi \\ x, & \text{otherwise} \end{cases} \tag{10}$$

The interested reader is referred to Funda (1991) for a more comprehensive treatment of both single and multiple contact motion restriction.

3.5 Providing Kinesthetic Feedback

It is well established that force reflection dramatically improves the sense of telepresence[5] in teleoperation (Goertz, 1963; Ferrell, 1966; Hannaford & Anderson, 1988). In fact, it has been shown that kinesthetic feedback can be at least as important as three-dimensional (3-D) visual information (Kilpatrick, 1976), and that, in some circumstances, force feedback alone can be more valuable than visual feedback alone (Ouh-young, Beard, & Brooks, 1989). Since the early experiments in extracting kinesthetic information from interaction with virtual objects (Noll, 1972) the problem has received considerable attention within the context of time-delayed telemanipulation. Approaches based both on kinematic as well as dynamic analyses of the virtual object interaction have been proposed (Noll, 1972; Buzan, 1989).

In the absence of a dynamic model of the remote environment, a teleprogramming system attempts to provide the operator with an approximate real-time kinesthetic sensation of the contact interactions between the simulated slave and the surrounding environment. Aside from reinforcing the operator's visual perception of the task environment, kinesthetic feedback also serves to help the operator maintain intuitive geometric correspondence between the commanded input and the display. Within the context of a kinematic simulation of the slave/environment interaction, a teleprogramming system provides the operator with a sense of kinesthetic teleperception by enforcing the same set of motion constraints S, which was used to restrict the contact motion of the simulated slave in Section 3.4, on the commanded motion of the master arm as well. The operator-supplied motion commands to the master arm $\mathbf{D}_m$ are therefore restricted using the same restriction operators introduced in Section 3.4 [Eqs. (9) and (10)], that is,

$$\mathbf{D}_m \rightarrow \boxed{S} \rightarrow \mathbf{D}'_m \tag{11}$$

and this restricted motion $\mathbf{D}'_m$ is used to drive the master arm. The master is therefore actively servoed to resist attempted motion in the constrained directions. This allows the operator holding the master arm to *kinesthetically* feel the impact of contacting a surface, reaching a corner, or pivoting about an edge.

While computed on the basis of a purely kinematic analysis, experimental results (Section 3.10) have shown that the resulting pseudo-force feedback provides a good approximation to actual force reflection for rigid-body environmental interactions and that it is extremely helpful to the operator during contact manipulation. An important feature of this approach is the fact that the same polyhedral analysis is reused for both contact motion restriction for the simulated slave and the kinesthetic feedback computation for the master arm.

3.6 A Priori Task Information

For the simulator software to correctly interpret the operator's actions, it may need to have some knowledge of the general goal of the task in progress or be aware of special characteristics of the task. For instance, a sequence of rapid contact changes may be interpreted either as noisy data (and thus get smoothed) or a purposeful action, such as tapping or scraping (in which case it should be kept intact). Similarly, a highly irregular path of an object during a sliding motion could be taken as unintended or it could correspond to a motion such as polishing or sanding. To disambiguate between such interpretations, the system needs additional information about the task, and in particular an indication of the types of expected primitive motions (e.g., pick and place, polishing, pounding). Other useful information

5. *Telepresence*—sensory illusion of being physically present in a remote environment.

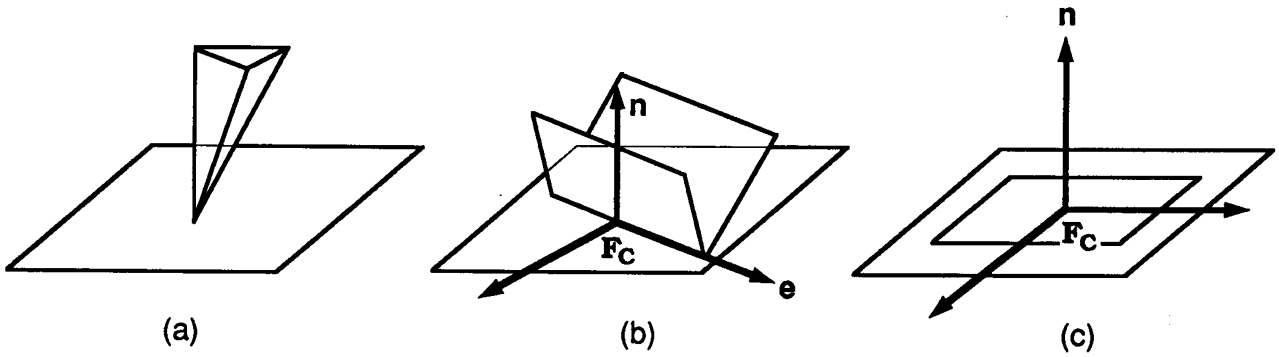

Figure 5. *Single contact pivoting.*

includes a list of the environmental objects and features that are expected to come into contact with the slave arm during the execution of the task, and an indication of the relevant relationships between the environmental objects involved in the task (e.g., which objects are rigidly attached to their support, which ones can be lifted or pushed, and so on). Information about the task may also be used by the system to automatically, on-line adjust the viewing angle, zoom, and other viewing parameters of the simulated remote environment, as well as to adjust the scaling of the commanded master device displacements or exerted forces into the slave space.

This task information can be gathered by querying the operator prior to task execution, by maintaining an on-line dialogue with the operator, or a combination of the two. The issues of specific task information content and representation have not yet been addressed in this work. Currently, the a priori task information is limited to a list of object pairs that are expected to come into contact during task execution (Section 3.3).

3.7 Generating Remote Slave Motion Commands

So far the operator can perform a task within the virtual environment of the graphic simulation by visually and kinesthetically interacting with the model. As the task specification proceeds, the operator's station software monitors the operator's activity in this simulated environment and translates this information into a stream of robot instructions. This is the central feature of a teleprogramming system and crucial to its success. The generated instructions must capture the essential features of the task in progress with sufficient accuracy to allow the specified actions to be reproduced at the remote site.

3.7.1 Approach. Two sources of information are available to the command generation module. The first consists of the low-level position and force trajectories, imparted by the operator, together with current contact state and motion mode information. This information is provided directly by the graphic simulator and is updated at each simulation step (Section 3.3). The other source of information is the task model as outlined in Section 3.6. This information guides the interpretation of the operator's actions and serves to resolve ambiguities. The output of the command generation module is a stream of elementary robot instructions, encompassing free space motion instructions and guarded and compliant motion primitives for contact motion and manipulation. The challenge, then, is to translate the observed operator's activity, including actions such as grasping, pushing, and part alignment, into sequences of guarded and compliant motion primitives.

As the operator's station based model of the remote environment is only approximate, the nature of the generated commands must reflect and accommodate the discrepancies between the modeled and the actual world to allow for stable and reliable execution at the remote site. Various control strategies for robust control of contact manipulation have been proposed over the past two

decades (Salisbury, 1980; Raibert & Craig, 1981; Hogan, 1985). We have in this work adopted the *hybrid position/force* control model (Raibert & Craig, 1981), where position (or velocity) is controlled along the *free* directions of motion, and force is controlled along the *constrained* d.o.f. of motion. A critical feature of the command generation process is the fact that the estimated model error tolerances (Section 3.1) are incorporated into the motion parameters of the appropriate (guarded) motions, so as to reduce the probability of execution failures due to small discrepancies between the model and the actual environment. It is also important to note that because the operator's station based simulation of the remote environment is kinematic in nature, the necessary dynamic parameters of the instructions (e.g., compliance forces, frictional parameters) cannot be given precisely. Instead, symbolic (or normalized numerical) values for these parameters are supplied to the slave, which in turn must substitute its current estimates of the actual values prior to execution. These estimates are based on the slave's previous interactions with the environment, that is, task history, and are derived from the local sensory readings at the remote site.

Elementary symbolic robot instructions are generated by analyzing the operator-supplied motion and force trajectories, elapsed time, and contact state changes between the simulated slave and the environment. A new sequence of commands, that is, an *execution environment,* is generated whenever the contact set C (i.e., the contact type(s) or multiplicity) changes or after the same contact state has persisted for t_{max} sec. The latter is done to avoid increasing the *lag time* between the master and the slave. t_{max} is a function of the bandwidth at which significant changes occur in the simulated environment, which in turn is related to the human neuromuscular bandwidth and should be on the order of 1 sec or less to provide for sufficiently accurate trajectory tracking.

3.7.2 The Command Language. Each execution environment is specified with respect to a Cartesian task frame $\mathcal{F}_T$. During contact motion, this frame is defined by the geometry of the current constraints acting on the motion of the robot. The axes of the frame are aligned with the dominant contact features and the origin is chosen to be coincident with the primary contact point, similarly as in the case of the contact frame $\mathcal{F}_C$ in Section 3.4. The system specifies the assignment of a new task frame by issuing the instruction

DefineTaskFrame (*name:ref-fm; org; x; y; z*)

where the italicized labels denote symbolic names of the corresponding entities. The task frame reference frame (*ref-fm*) is either a global world frame (in which case the task frame remains fixed with respect to the world during the motion and is said to be *static*) or the slave end-effector frame (in which case the task frame moves along with the slave robot and is said to be *dynamic*). The numeric values of the defining vectors are supplied by the graphical simulator and relayed to the remote site using the statement

DefineVector (*name*; ⟨x,y,z⟩:*ref-fm*)

Once defined, the axes of the task frame must be designated as either force or position controlled, which in turn defines the hybrid control *selection matrix.* Force and position trajectories can then be specified along the force and position controlled axes, respectively. In addition, force (velocity) preloads can be specified in the position (force) controlled directions for applications such as pushing (tracking). For example, during an edge/face contact, the following sequence of commands would be generated to execute a simple sliding motion in the direction of this edge (see Fig. 6)

UseFrame (F1)
AssignMode (P, P, F, F, P, P)
Force (⟨0,0, $-f_s$⟩; ⟨0,0,0⟩)
Slide (t; ⟨0,d,0⟩)

where explicit definitions of the task frame axes have been omitted for brevity. f_s denotes a positive force (given as a normalized numerical value), which is sufficient to ensure sustained contact during the sliding motion, and the time parameter t imposes a velocity requirement onto the motion.

To prevent execution errors due to the model uncertainties, the following safeguards are provided by the language

GuardForce (⟨f_x, f_y, f_z⟩; ⟨τ_x, τ_y, τ_z⟩)
GuardVelocity (⟨v_x, v_y, v_z⟩; ⟨ω_x, ω_y, ω_z⟩)

The slave should stop the motion if any of the specified guard conditions become true. As an example, a pivot-

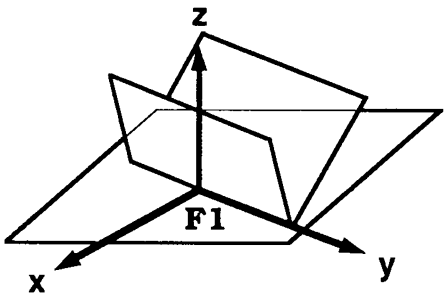

Figure 6. *Example 1—single contact sliding.*

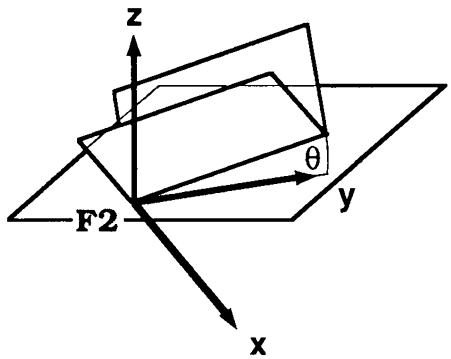

Figure 7. *Example 2—single contact pivoting.*

ing motion about the contact point transitioning from a vertex/face to an edge/face contact is encoded as follows (see Fig. 7):

UseFrame (F2)
AssignMode (F, F, F, P, P, P)
Force ($\langle 0,0,-f_s \rangle$; $\langle 0,0,0 \rangle$)
GuardForce ($\langle 0,0,0 \rangle$; $\langle l \cdot f_s, 0,0 \rangle$)
Pivot (t; $\langle -(\theta + \epsilon_o), 0,0 \rangle$)
AssignMode (F, F, F, F, P, F)

Here the task frame has been aligned with the impending edge contact, l denotes the edge length, and $l \cdot f_s$ is the expected motion termination torque about the x-axis of the task frame. Note that the rotational displacement θ is increased by the upper bound on the orientational uncertainty ϵ_o in the model. Also note that the mode information (i.e., selection matrix) is updated, following the rotation into contact (contact stiction is assumed).

A number of other factors need to be considered in the command generation process. The interested reader is referred to Funda (1991) for a more comprehensive treatment of the problem.

3.8 The Remote Slave Robot

The communication between the command stream generator (operator's station) and the remote site proceeds in terms of execution environment packets. The slave workcell receives each execution environment a transmission delay τ after it was sent and places it into its *execution queue*. Execution at the slave site then proceeds by parsing and translating successive execution environments, substituting numerical values for the symbolically specified dynamic parameters (e.g., frictional parameters and compliance force levels), and passing the resulting code to the local controller for execution.

3.8.1 Parsing and Buffering. In our experimental system the parsing/translating stage is performed by a simple **yacc**-based interpreter, which parses the incoming command strings and generates the corresponding RCI+ instructions. RCI+ here refers to a Cartesian hybrid position/force controller, which was built on top of the low-level RCI robot interface (Hayward, 1983). Due to the simplicity of the symbolic command language (Section 3.7.2) and the fact that it was designed to go hand-in-hand with RCI+, the translation process is efficient and straightforward. Some of the instructions (DefineVector, DefineTaskFrame) simply update the *symbol table* of currently defined vectors and coordinate frames, which the remote controller software uses during execution. Others set various parameters in the hybrid controller directly (e.g., AssignMode, Move) or provide the necessary symbolic information to compute actual parameter values explicitly (e.g., Force, GuardForce).

During execution care must be taken to avoid increasing the *lag time* η between the master and the slave manipulators as the task proceeds. To ensure constant lag, the remote controller employs a combination of double-buffering and an initial holding time ht_0. The controller places each translated execution environment into a *command buffer*. Two such buffers are maintained—while one is being executed, the other one is being constructed by translating the next execution environment. Together with a holding time of $ht_0 = 2t_{max}$ it can be shown that

the lag time is constant and bounded by $\eta = \tau + ht_0$ (Funda, 1991).

3.8.2 Control of the Slave. The symbolic instructions arriving at the slave site are based on an imperfect model of the actual environment. Despite the fact that the critical motion parameters (e.g., distances to surfaces or edges) have been computed so as to account for the estimated uncertainties in the modeling, sensing and control errors during execution may still cause execution failures. Consequently, a robust controller and integrated real-time sensing capability are needed to handle contact interactions with an imperfectly known environment. The slave execution process must proceed as a high-bandwidth local feedback loop with sensory input participating in the real-time control decisions.

In our system we currently make use of an instrumented compliant wrist sensor (Lindsay, 1991), which provides passive compliance at the end-effector as well as real-time 6 d.o.f. contact force information. This sensor is integrated with a Cartesian level hybrid force/position controller, which provides for compliant and locally adaptive response in contact manipulation. The combination of passive compliance provided by the wrist sensor and active compliance provided by the control algorithm allows the slave robot to perform guarded and compliant contact motions stably and reliably.

3.9 Error Handling

Despite a robust execution controller at the slave site, things still may (and will!) go wrong. Some of the common execution errors are not reaching an expected motion terminating condition (force, distance), hitting an obstacle in the workspace, stopping prematurely during guarded motions, jamming, etc. The slave should be able to detect most of these error conditions by monitoring its position, velocity, end-effector force, and motor torques. On detecting such a condition, the slave must fail safely and gather as much relevant information about the error state as possible. This information is then sent to the operator's station. Because of the transmission and other delays between the operator's station and the remote site, the operator learns about an error condition at the slave site $\eta + \tau$ sec later. During this time, the operator had continued with the task and possibly modified the simulated environment. The error packet arriving from the slave must therefore contain sufficient information to restore the simulated environment to the error state and present to the operator the critical remote site sensory data. If this should not suffice for the operator to understand the nature of the problem at the remote site, the operator should be able to initiate exploratory actions at the slave workcell (e.g., request additional camera views of the contact area) and gather additional information. Once the operator has determined the cause of the error, corrective actions can be specified to recover from the error and continue with the task. Therefore, by keeping the operator in the control loop, the system eliminates the need for exception and error handlers to be preprogrammed off-line.

Moreover, the operator's station model could be updated continuously as the task progresses. While interacting with the environment, the slave manipulator comes into contact with various objects and features in the environment and can thus precisely determine their position and orientation relative to its local reference coordinates. As the task progresses, this information can be accumulated and used to provide for on-line refinement of the operator's station-based model. However, as this information will normally be noisy, error prone, and local, care must be taken in propagating this local corrective information through the model.

In our current experimental system the state of the environment is completely specified by the configuration of the slave robot and the position of any grasped object(s) with respect to the end-effector. On detecting an error, the slave controller therefore informs the operator's station of the error condition and reports its current (end-effector) position. The operator's station software in turn alerts the operator through audio (synthesized voice message) and visual (display flashing) means, and commences the error recovery procedure. The world model is updated to reflect the current remote environment configuration and the master arm is reindexed. No continuous on-line model updating or refinement is currently done.

3.10 The Experimental System and Results

3.10.1 The Experimental System. The experimental teleprogramming platform, which was developed at the GRASP Laboratory in order to test the teleprogramming concept, is shown in Figure 8. The system was christened MERIONETTE for Model-based EditoR for Interactive ON-linE Teleprogramming in Time-delayed Environments. In Figure 8, the right-hand display at the operator's station shows the real-time graphic simulation, whereas the left-hand display provides the delayed actual video from the remote site.

The hardware architecture of MERIONETTE is shown in Figure 9. The master arm in our set-up is a Unimation PUMA 250 manipulator with a 6-axis force/torque sensor mounted at its tip providing for Cartesian motion input. Hardware control for the master is provided by the PC-bus based Modular Motor Control System (MMCS) (Corke, 1989). The computational engine of the operator's station subsystem is JIFFE—a 20 Mflop VME-based floating point processor (Andersson, 1989), which runs both the low-level joint servo code (500 Hz), as well as the Cartesian level servo code and kinesthetic feedback computation of Section 3.5 (30 Hz). JIFFE communicates with the host (Sun 3/160) via shared memory and with the graphic workstation (Iris 4D-25) via the Sun and a TCP/IP socket connection. The software modeling environment for 3-D manipulation of articulated figures was provided by the Computer Graphics Laboratory at the University of Pennsylvania (Phillips & Badler, 1988). Also interfaced to the Iris is a simple audio playback system, which allows for run-time playback of prerecorded digitized warning and information messages under software control.

The remote slave manipulator in our experimental system is a PUMA 560, which is interfaced to a MicroVax II, running the RCI+ programming environment (Section 3.8). The low-level joint servo control is accomplished via the standard Unimate controller ($\approx$ 1000 Hz), while the MicroVax provides for Cartesian control of the manipulator ($\approx$ 35 Hz). A 6-axis instrumented compliant wrist is used as the remote sensing device (Section 3.8). The communication between

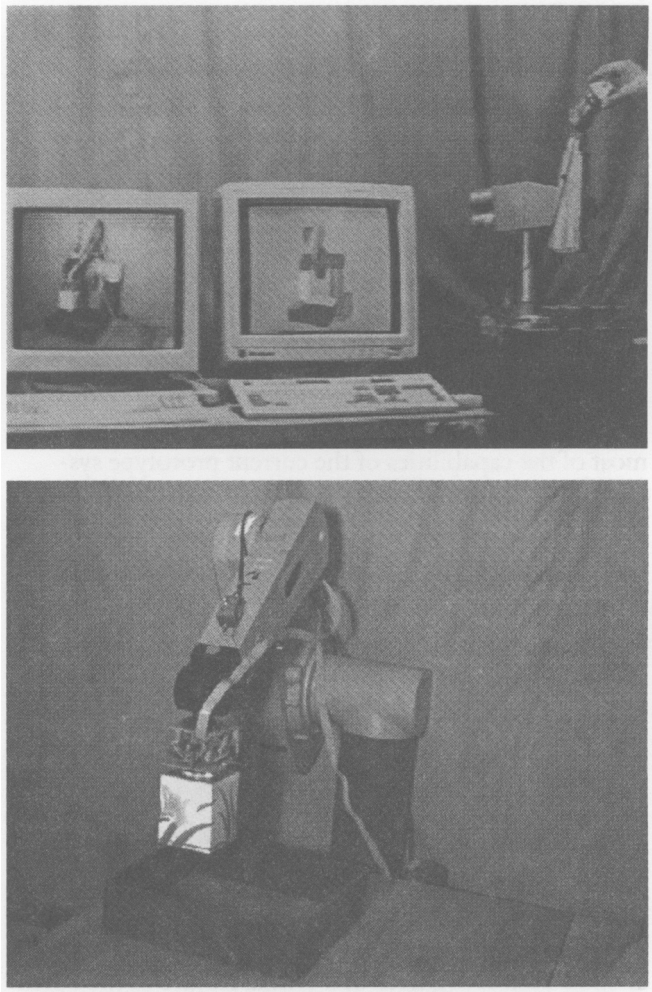

Figure 8. *MERIONETTE: operator's station (top) and remote site (bottom).*

the operator's station and the slave manipulator is again facilitated by a bidirectional UNIX TCP socket link, which can be programmed to simulate an arbitrary transmission delay.

3.10.2 Results. So far the experimental system's performance has been tested on the simple task of sliding around the inside of an open box (Fig. 8). The box exploration task was chosen because of its highly interactive nature. The operator spends most of the time in contact with the environment, sliding around the box bottom, following edges, reaching corners, etc. Despite its simplicity, this testing paradigm allows us to exercise

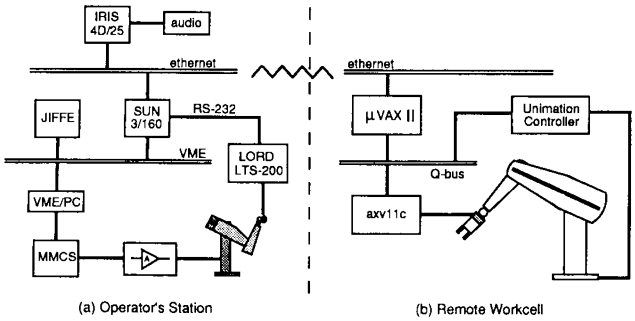

Figure 9. *The hardware architecture of MERIONETTE.*

most of the capabilities of the current prototype system—the kinesthetic operator–environment interaction, on-line low-level symbolic command generation, remote site execution, as well as the preliminary implementation of the error recovery facility. Transmission delays up to 3 sec have been tested. To date no rigorous quantitative studies have been performed using the system.

Preliminary experiments have confirmed that the teleprogramming control methodology indeed allows the remote slave to be controlled smoothly and efficiently in the presence of a substantial feedback delay. The operator is able to "feel" around the box, with the remote slave following along a lag time η later. The kinesthetic interaction with the simulated environment feels natural and has been recognized as a crucial feature of the operator–simulator interface, dramatically contributing to the operator's ability to perform contact tasks confidently and efficiently. The tests have also confirmed that at least for the case of kinesthetic exploratory activity, the system is capable of automatically generating the corresponding robot instructions and that the resulting symbolic program provides a sufficiently detailed description of the slave's activity to ensure stable and robust execution at the remote site. As expected, execution errors at the remote site relate primarily to the unmodeled static and dynamic effects of real world contact interaction. In particular, friction forces are occasionally mistaken for terminating conditions in guarded moves, or the estimated compliance forces are insufficient to maintain contact during a compliant motion. However, because all contact motions contain explicit terminating conditions (e.g., force, distance, time), error conditions are easily detected by the slave controller. Experiments have shown that on detecting an error, the preliminary error recovery mechanism allows the operator to recover and successfully continue with the task in most cases. Research is currently under way to provide for a more comprehensive error recovery, where the slave workcell would be able to gather more detailed information about its error configuration through local exploratory procedures. Issues related to on-line model refinement are also being investigated.

In summary, whereas much more work remains to be done, the preliminary testing confirmed the validity and effectiveness of the teleprogramming control paradigm. Remote site autonomy at the level of $nt > 100$ sec was successfully achieved with the box exploration task, verifying the feasibility of near-optimal teleprogramming remote control as postulated in Section 2. Research continues on refining and extending the theoretical basis and parallel implementational and experimental efforts are being made to verify the proposed ideas.

4 Conclusion and Future Directions

The teleprogramming concept described in this paper integrates a number of existing telerobotic concepts with new ideas in operator–model interaction and automatic robot programming into a new telerobotic architecture. The key focus and contribution of this work is in developing and demonstrating a means of automatic interpretation and symbolic encoding of the operator-supplied 6 d.o.f. task specification in a virtual model. The combination of compliant control paradigm as the framework for the generated instructions, incorporation of estimated model errors directly into the command parameters, and sensor-driven high-bandwidth execution of primitive actions at the remote site accounts for significantly increased reliability of execution by the slave robot. Moreover, the symbolic encoding of the task primitives represents a significant compression of the unprocessed control signals supplied by the operator.

The key feature of a teleprogramming system is that it is capable of providing for efficient control of a remote robotic system while relying on a relatively modest amount of remote site autonomy. Instead of increasing

the complexity of the remote robot controller to compensate for modeling errors, the teleprogramming approach attempts to build more "intelligence" into the control signals (the robot instructions) that are being sent to the remote slave. As the task specification proceeds, the flow of control is interrupted only when errors occur at the remote site. The applicability and effectiveness of the teleprogramming control paradigm in terms of the amount of feedback delay are therefore limited only by the constraints on the amount of autonomy that robotic systems can currently provide and by the amount of time and work that the operator can tolerate losing when an error is reported and the operator is placed back to the point in the task execution where the error occurred. The latter puts a limit on the acceptable lag time between the master and the slave, and thus the length of tolerable feedback delays. An error recovery mechanism, whereby the system would monitor the operator's corrective actions and resume autonomous execution as soon as it recognized a state, which was part of the original task specification, would significantly improve the overall system efficiency, as well as operator satisfaction. Research is currently underway to assess the feasibility of this and other error recovery schemes.

Taking a slightly different view of the teleprogramming paradigm, the operator's station can be thought of as a stand-alone facility for automatic generation of robot programs, based on operator's task specification in a model-based virtual environment. The generated programs can be executed immediately, delayed in time, or simply stored for later execution. Viewed in this manner, the operator's station becomes a virtual environment-based, kinesthetic and visual robot program editor. Moreover, the operator could use such an editor to control a number of robotic devices, either remote or nearby, by decoupling the task specification rate from the execution rate. Thus, control of multiple slow devices could be accomplished by interleaving the corresponding task specifications in the kinesthetic graphic editor, which could proceed at a much faster rate. In case of an execution error on a specific device, the operator could (according to a priority scheme) attend to this device and resume task specification for other devices, or stand by while execution at the slave site proceeds. This type of control relieves the operator of continuous interaction with the operator's station and provides the basis for more general forms of supervisory control of robotic devices.

Recent experimental results with our prototype teleprogramming platform have confirmed the feasibility and effectiveness of the teleprogramming concept and provided strong encouragement for future refinements and extensions to the existing methodology. Although the preliminary experiments have demonstrated only very basic functionality, it is important to note that much of the manipulative capability required in space and underwater applications (e.g., satellite module replacement, locating an eye-bolt and inserting a hook) relates to the ability to kinesthetically explore the surroundings and locate various features, using sequences of primitive contact motions of the type demonstrated in our system. Future work will be aimed at increasing the dexterity and versatility of the current system to allow for broader applicability in various situations and environments.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant BCS-89-01352, "Model-Based Teleoperation in the Presence of Delay." Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Anderson, R. J., & Spong, M. W. (1988). Bilateral control of teleoperators with time delay. *Proceedings of the IEEE International Conference on Robotics and Automation,* 131–137.

Andersson, R. L. (1989). Computer architectures for robot control: A comparison and a new processor delivering 20 real Mflops. *Proceedings of the IEEE International Conference on Robotics and Automation,* 1162–1167.

Bejczy, A. K., & Kim, W. S. (1990). Predictive displays and shared compliance control for time-delayed telemanipulation. *IEEE International Worksop on Intelligent Robots and Systems,* Ibaraki, Japan.

Bejczy, A. K., Kim, W. S., & Venema, S. C. (1990). The

phantom robot: Predictive displays for teleoperation with time delays. *Proceedings of the IEEE International Conference on Robotics and Automation,* 546–551.

Besl, P. J., & Jain, R. C. (1985). Three-dimensional object recognition. *ACM Computing Surveys, 17*(1).

Black, J. H. (1971). *Factorial study of remote manipulation with transmission time delay.* Master's thesis, Department of Mechanical Engineering, MIT, Cambridge, MA.

Bolle, R. M., & Cooper, D. B. (1986). On optimally combining pieces of information, with application to estimating 3-D complex-object position from range data. *IEEE Transactions On Pattern Analysis and Machine Intelligence, 8*(5).

Brooks, T. L. (1990). Telerobotic response requirements. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* 113–120.

Buzan, F. T. (1989). *Control of telemanipulators with time delay: A predictive operator aid with force feedback.* Ph.D. thesis, Massachusetts Institute of Technology, Canbridge, MA.

Corke, P. I. (1989). *A new approach to laboratory motor control: The modular motor control system.* Tech. Rep. No. MS-CIS-89-17. Philadelphia: University of Pennsylvania.

Ferrell, W. R. (1965). Remote manipulation with transmission delay. *IEEE Transactions on Human Factors in Electronics, 6*(1).

Ferrell, W. R. (1966). Delayed force feedback. *IEEE Transactions on Human Factors in Electronics,* 449–455.

Ferrell, W. R., & Sheridan, T. B. (1967). Supervisory control of remote manipulation. *IEEE Spectrum, 4*(10), 81–88.

Funda, J. (1991). *Teleprogramming: Towards delay-invariant remote manipulation.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Gilbert, E. G., Johnson, D. W., & Keerthi, S. S. (1987). A fast procedure for computing the distance between objects in three-space. *Proceedings of the IEEE International Conference on Robotics and Automation.*

Goertz, R. C. (1963). Manipulators used for handling radioactive materials. In E. M. Bennett (Ed.), *Human factors in technology* (ch. 27). New York: McGraw-Hill.

Hannaford, B. (1989). A design framework for teleoperators with kinesthetic feedback. *IEEE Transactions on Robotics and Automation, 5*(4).

Hannaford, B., & Anderson, R. (1988). Experimental and simulation studies of hard contact in force reflecting teleoperation. *Proceedings of IEEE International Conference on Robotics and Automation,* (1), 584–589.

Hayward, V. (1983). *RCCL User's Manual.* Tech. Rep. No. TR-EE-83-46. Purdue: Purdue University.

Hirzinger, G., Heindl, J., & Landzettel, K. (1989). Predictive and knowledge-based telerobotic control concepts. *Proceedings of the IEEE International Conference on Robotics and Automation,* 1768–1777.

Hogan, N. (1985). Impedance control: An approach to manipulation. *ASME Journal of Dynamic Systems, Measurement and Control, 107,* parts I-III, 1–24.

Kilpatrick, J. P. (1976). *The use of kinesthetic supplement in an interactive system.* Ph.D. thesis, University of North Carolina, Chapel Hill, NC.

Lindsay, T. (1991). *Design of a tool-surrounding compliant instrumented wrist.* Tech. Rep. No. MS-CIS-91-30. Philadelphia: University of Pennsylvania.

Noll, M. A. (1972). Man-machine tactile communications. *SID Journal.*

Noyes, M., & Sheridan, T. B. (1984). A novel predictor for telemanipulation through a time delay. *Proceedings of the 20th Annual Conference on Manual Control.*

Ouh-young, M., Beard, D., & Brooks, F. (1989). Force display performs better than visual display in a simple 6-D docking task. *Proceedings of the IEEE International Conference on Robotics and Automation,* 1462–1466.

Phillips, C. B., & Badler, N. I. (1988). Jack: A toolkit for manipulating articulated figures. *Proceedings of ACM/SIGGRAPH Symposium on User Interface Software.*

Raibert, M. H., & Craig, J. J. (1981). Hybrid position/force control of manipulators. *ASME Journal of Dynamic Systems, Measurement and Control,* 126–133.

Raju, G. J. (1988). *Operator adjustable impedance in bilateral remote manipulation.* Ph.D. thesis, Department of Mechanical Engineering, MIT, Cambridge, MA.

Salisbury, J. K. (1980). Active stiffness control of a manipulator in Cartesian coordinates. *19th IEEE Conference on Decision and Control.*

Sawada, C., Ishikawa, H., Kawase, K., & Takata, M. (1989). Specification and generation of a motion path for compliant motion. *Proceedings of IEEE International Conference on Robotics and Automation,* 808–815.

Sheridan, T. B. (1990). *Telerobotics and human supervisory control.* In press.

Srihari, S. N. (1981). Representation of three-dimensional digital images. *ACM Computing Surveys, 13*(4).

Stark, L. W., Kim, W. S., & Tendick, F. (1988). Cooperative control in telerobotics. *Proceedings of the IEEE International Conference on Robotics and Automation,* 593–595.

Yokokohji, Y., & Yoshikawa, T. (1990). Bilateral control of master-slave manipulators for ideal kinesthetic coupling. *IEEE International Workshop on Intelligent Robots and Systems,* 355–362.