Craig P. Sayers
and
Richard P. Paul
General Robotics and Active Sensory Perception Laboratory
Department of Computer and Information Science
The University of Pennsylvania
Philadelphia, Pennsylvania 19104
sayers@grip.cis.upenn.edu
http://www.cis.upenn.edu/~sayers/home.htm/

An Operator Interface for Teleprogramming Employing Synthetic Fixtures

Abstract

In the teleprogramming system an operator is presented with a virtual reality representation of a remote environment. The operator's interaction within that virtual environment is observed and translated into a sequence of robot program instructions for transmission to, and execution by, a robot at the remote site. In this paper we focus on operator interaction with the master station of the teleprogramming system. The use of synthetic fixtures to provide force and visual clues to assist the operator in performing tasks with speed and precision is discussed. It is suggested that, at least in some situations, it is both necessary and desirable to trade off realism for improved task performance. The difficulty of coping with exceptional conditions and, in particular, uncertainty in the world model used to generate the virtual environment is described and the operator interface for diagnosing and resolving errors is presented. An overview is also given of both the hardware and software used to implement the master station for the teleprogramming system.

1 Introduction

The motivation for this research comes from a desire to perform manual tasks using a robot manipulator mounted on an unmanned untethered submersible. Without a tether we must work within the constraints of an acoustic communications link—typical bit rates being on the order of 10 kbits/sec while round trip time delays are on the order of 10 sec (Dunbar et al., 1990).

There are two extreme approaches. One alternative is to perform tasks in a highly automated fashion, providing the subsea robot with sufficient sensory and processing ability to enable it to work fully autonomously—essentially taking the knowledge and experience of a human operator and encoding it into the remote vehicle. Such an approach is undesirable for two reasons: it requires a very expensive and sophisticated slave robot and it is difficult to provide autonomous solutions that can cope with all unexpected events. A second alternative is to not provide any automation and have a very simple slave robot that is controlled in a teleoperative mode (Vertut & Coiffet, 1986) by an operator either ashore or on a surface vessel. In this case a human operator provides all of the system's memory and reasoning. The difficulty with this method is that the large roundtrip time delay makes it impossible to perform tasks that require rapid responses to sensory input. Even those tasks that are possible may be performed only very slowly with operators tending to adopt a "move and wait" strategy (Ferrell & Sheridan, 1967).

PRESENCE, *Vol. 3, No. 4, Fall 1994, pp. 309–320*

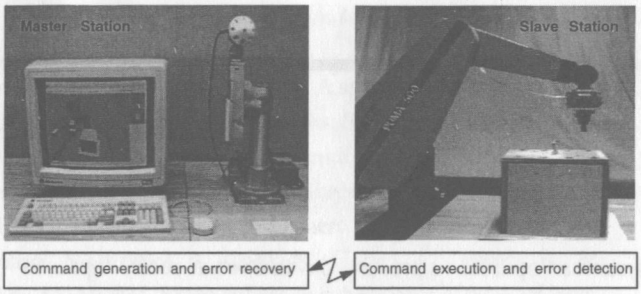

Figure 1. *An overview of the teleprogramming system showing the current experimental in-air setup. Communication between master and slave sites occurs over a low-bandwidth, delayed communications link.*

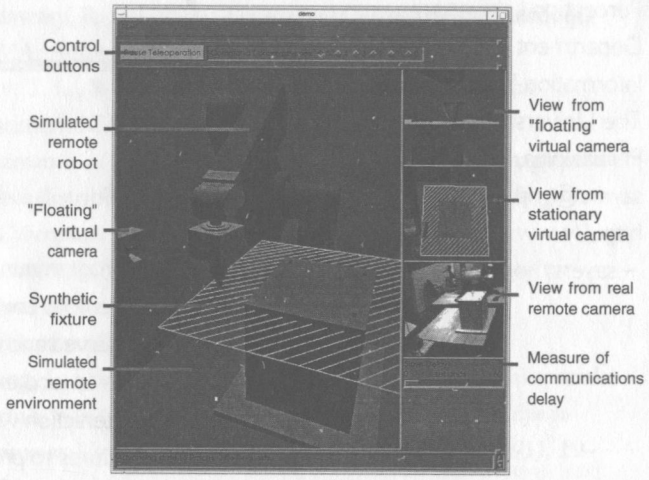

Figure 2. *The visual interface presented to the operator during teleprogramming. Multiple views are available of the simulated remote site. The control buttons may be activated either with the mouse or by foot switches.*

The teleprogramming system (see Fig. 1) developed by Paul, Funda and Lindsay falls somewhere between these two extremes (Funda, 1991; Funda, Lindsay, & Paul, 1992; Lindsay, 1992). The operator is provided with a virtual reality representation of the remote site that provides immediate visual and kinesthetic feedback. The system observes the operator's interaction within that simulated world and generates a symbolic, error-tolerant, command stream for transmission to and execution by the remote slave manipulator. In this case only knowledge that is necessary for the current situation is passed from the human to the slave robot. Similar approaches have been developed for teleoperation in space (Bejczy, Venema, & Kim, 1990; Hirzinger, 1993; Kotoku, Tanie, & Fujikawa, 1990; Machida, Toda, & Iwata, 1990; Sheridan, 1993).

This paper focuses on the master station of the teleprogramming system. It describes how the force and visual clues generated by synthetic fixtures are used to assist the operator in performing tasks with speed and precision. Consideration is given to cases where the world model used to generate the virtual environment is inaccurate and an operator interface for diagnosing and resolving execution errors is presented.

2 Overview of the Master Station

The teleprogramming master station presents the operator with a graphic interface with which he or she can create, view, and modify the simulated slave site (see Fig. 2). Once a teleprogramming task has begun the operator's primary interaction with the system is through the use of a 6 degree-of-freedom (DOF) master manipulator. This serves as both an input device (sensing where the operator wants to move) and an output device (giving kinesthetic force clues to the operator). This interaction is aided by the use of synthetic fixtures that provide both force and visual clues (see Section 3). As the operator performs the task his or her actions are monitored and translated into a sequence of robot program instructions (see Section 5) for transmission to the remote slave robot. By comparing command expectations predicted by the master station with those results actually recorded during task execution, the slave is able to detect execution errors. These may be due to uncertainty in the world model, positioning errors, or other unexpected events. When such an exceptional condition is detected the slave robot pauses execution and advises the master station. The operator must then diagnose the problem and take corrective action (see Section 6), generating new commands for the slave manipulator.

3 Synthetic Fixtures

A number of authors have described the use of computer-generated visual or force clues to assist operators in task performance. For example, it is possible to

add shadows and textures (Wanger, Ferwerda, & Greenberg, 1992), to superimpose visual enhancements (Kim & Stark, 1989; Oyama, Tsunemoto, Tachi, & Inoue, 1993), and generate predictive displays (Bejczy et al., 1990; Hirzinger, 1993; Kotoku et al., 1990; Sheridan, 1993). Force clues have been used to create synthetic input/output devices (Hollis & Salcudean, 1993), to guide chemists in docking molecules (Ouh-young, Beard, & Brooks, 1989), to simulate collisions with objects (Anderson, 1993; Millman, Stanley, Grafing, & Colgate, 1992; Kim & Bejczy, 1991; Tanie & Kotoku, 1993), to repel the end-effector from obstacles in the environment (Bruno & Morgenthaler, 1991), to constrain end-effector motion in one or more degrees or freedom (Unruh, Faddis, & Barr, 1992), and to confine the boundaries of the end-effector trajectory (Rosenberg, 1992).

Synthetic fixtures (Sayers, Paul, & Mintz, 1992; Sayers & Paul, 1993) provide an operator with task-dependent and context-sensitive visual and force clues. The system does not attempt to provide realism. Instead the intention is to provide the operator with those force and visual clues that can best aid him or her in task performance. For example, consider the case in which the operator moves the end-effector toward a surface. If it is appropriate for the surface to be contacted then the system activates an attractive fixture to assist the operator in moving to and then maintaining contact with the surface. Alternatively, if surface contact was inappropriate then a repulsive fixture would be employed. The system does not just react to the operator's input, instead it attempts to predict and then actively assist his or her actions.

Imagine the hypothetical example of a virtual reality simulation of a computer keyboard. If the intention were to give the user the illusion that they were using a particular real keyboard then it would be important to accurately simulate forces. However, if our task is merely to help a user type then it would be quite acceptable just to have every key feel approximately "key-like." Furthermore, by appropriate use of force clues the operator could be provided with much more information than just whether or not a key was being pressed. For example, the force required to activate each key could be varied based on

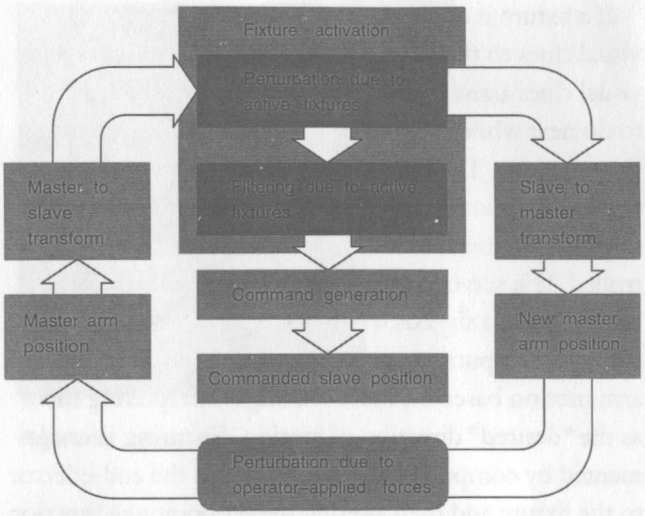

Figure 3. *A logical overview of the fixturing implementation.*

- Task knowledge/past history—which keys should/should not be pressed at this time.
- Current configuration—which keys are currently being activated

Thus, by providing force clues that are task and configuration dependent, the system can take a very active role in assisting the operator in completing a given task.

Synthetic fixtures are analogous to the "snap" family of commands used in computer-aided design programs (Sutherland, 1963; Autocad, 1989; Bier & Stone, 1986), increasing both the speed and precision with which an operator can work.

3.1 Operation

Fixturing operates in three stages (see Fig. 3). The first stage is activation, where a fixture decides whether or not it should operate. These activation decisions should be made automatically—if operators had to manually select, activate, and deactivate fixtures then much of the time saved by using fixtures would be lost. The system should attempt to predict the operator's expected actions and then, based on the task, activate those fixtures that can best assist the operator. A simple scheme is to have fixtures activate whenever an appropriate end-effector tool is nearer than some prespecified distance.

If a fixture is active then it provides both force and visual clues to the operator. During this second stage visual clues assist the operator in deciding which actions to do next while force clues assist in actually performing those actions. These force clues are generated by perturbing the motion of the master arm. In the current implementation the master manipulator is position controlled. It is servoed by reading the force the operator exerts (using a six-axis wrist-mounted force/torque sensor) and computing a new Cartesian set point for the arm motion based on those readings, interpreting them as the "desired" direction of motion. Fixturing is implemented by computing the distance from the end-effector to the fixture and then altering the set point as a function of that distance.

Fixtures are activated/deactivated when the end-effector is sufficiently far away as to apply no apparent force to the operator. This avoids problems with the end-effector appearing to jerk as fixtures are autonomously turned on and off.

If the operator allows the force clues to guide him or her to a position very close to some feature, then it is assumed that the operator intended to move exactly to that feature. The commanded slave position is then filtered to reflect this assumption. In this way the system combines the operator's relatively imprecise motions with a little task knowledge to generate precise commands. This is analogous to the "snap" commands used in CAD programs. It has the effect of preventing positional errors at the operator station from "flowing through" to the rest of the system.

3.2 Terminology

We define most fixtures to be of type X–Y where X is some feature on the end-effector and Y is some feature in the environment. Unless otherwise stated it is assumed that the system should aid the operator in bringing feature X on to feature Y. For example, a *point–point fixture* aids the operator in moving a point on the end effector to a point in space while a *closest-surface–surface fixture* assists in moving the closest surface (plane of finite extent) on a polygonal end-effector into contact with a particular surface in the world.

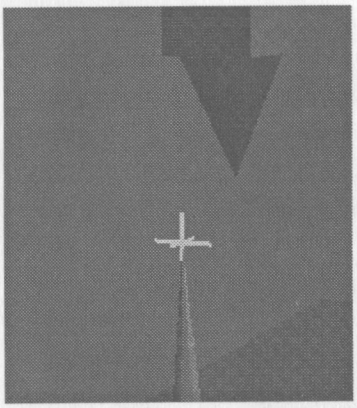

Figure 4. *A point–point fixture.*

3.3 Command Fixtures

Synthetic fixtures enable tools to be controlled in an intuitive manner without requiring additional physical controls. Consider the case in which the end-effector is a tool that must be turned on and off. One possibility for controlling this tool would be to add physical controls such as push-buttons or foot-switches. However, it would be preferable if we could merely observe the operator's motions and then automatically generate commands to turn the tool on and off at appropriate times. This is complicated since the operator's motions are often ambiguous. However, by using active force and visual clues much of this ambiguity can be resolved. The basic idea is to place a virtual barrier around the position in which the tool should be used. By pressing through this barrier the operator clearly indicates the intention to activate the tool. The barrier is typically implemented as a closest-surface–surface fixture that calls a function whenever the end-effector penetrates it.

3.4 Example Applications

To better describe the nature of fixtures we present some specific examples ranging from simple single-fixture applications to more complex multifixture scenarios.

3.4.1 A Point–Point Fixture. Here a *point–point fixture* (Fig. 4) is used to aid in bringing the tip of the end effector (the dark cone) toward a point in the world. The operator has brought the end-effector sufficiently close for the fixture to activate causing the bright star to

appear on the screen. This visual indication of activation is important. It gives the operator some idea of what subsequent actions the system is expecting and it typically precedes the use of any perceptible force clues. This gives the operator an opportunity to take corrective action if the system's "guess" as to his or her desired actions is incorrect.

With the end-effector still some distance from the fixture the operator feels a slight force pulling him or her toward the fixturing point. The clues are generated by the equation:

$$\mathbf{p}'(e) = \begin{cases} \mathbf{p}(e) - \mathbf{h}(e)W(|\mathbf{h}(e)|) & \text{if } |\mathbf{h}(e)| < T_a \\ \mathbf{p}(e) & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{p}(e)$ and $\mathbf{p}'(e)$ are the locations of the point on the end-effector before and after perturbation by the fixture, $\mathbf{h}(e)$ is the displacement of the end-effector point from the fixture, T_a is the fixture activation distance, and $W(d)$ is the fixture weighting function—a typical example being

$$W(d) = \frac{1}{1 + k_1 d^{k_2}} \quad (2)$$

where k_1 and k_2 are constants determining the "feel" of the fixture. The effect is that the operator feels a force pulling him or her toward the fixture point. This force increases as the end-effector is brought closer to the fixture. Typically these force clues are sufficiently strong that the operator can feel and interact with them but not so strong that motion in an alternative direction is prevented. In this way fixtures guide, but never completely constrain, the operator.

3.4.2 A Point–Path Fixture. A *point–path fixture* (Fig. 5) is used to aid in bringing the tip of the end effector to a path composed of line segments and arcs. Moving the end-effector so as to trace along this path requires no effort while moving away from the path requires that the operator overcome a distinct resistance. In this example the operator is able to accurately trace along a 3-D path despite the fact that they may only be working with a 2-D visual display. The limited depth information available to the operator from visual means is supplemented by that available through force clues.

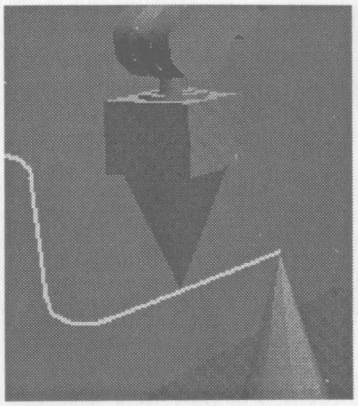

Figure 5. *A point–path fixture.*

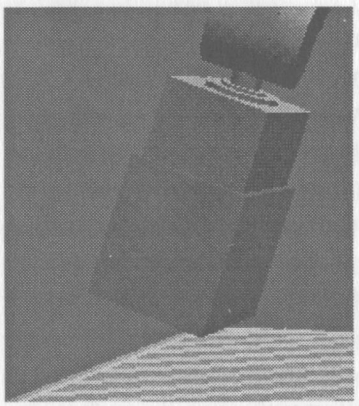

Figure 6. *A closest-surface–surface fixture.*

They provide the operator with information that may be either unavailable or difficult to perceive from other sources.

3.4.3 A Closest-Surface–Surface Fixture. In this example a *closest-surface–surface fixture* (Fig. 6) is used to aid in bringing the end effector (the dark box) into a face–face contact with a surface in the world. The operator has brought the end-effector sufficiently close for the fixture to activate causing the surface to be highlighted (the hashed lines). At this point the operator feels a slight force pulling the end-effector down and a slight moment rotating it toward a flat face–face contact. If the operator chooses to move closer to the surface then the magnitude of these clues will increase, guiding the end-effector to a perfect contact.

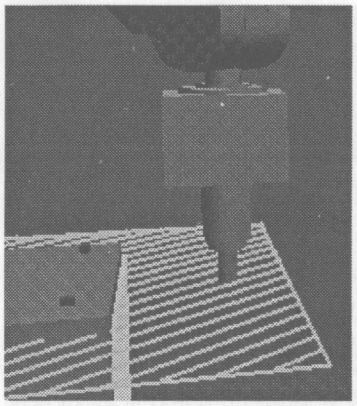

Figure 7. *Multiple fixtures for box interaction.*

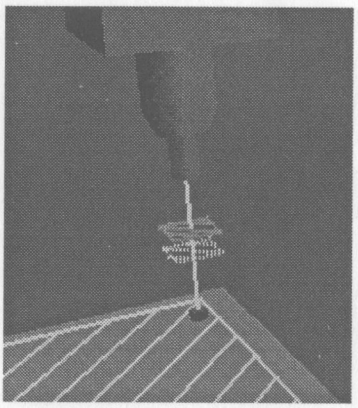

Figure 8. *Multiple fixtures for bolting/unbolting.*

Surface fixtures differ from a simulated virtual wall in two respects. First, they provide force clues before contact occurs. Rather than just telling the operator when they have hit a surface they actively guide the end-effector toward flat face–face contacts. The feeling for the operator is similar to moving a magnetized block around in a frictionless metal world. Second, the force clues are strongest when the end-effector is very close to a face–face contact and decrease as the end-effector moves away from this position. If the operator chooses to push hard enough to force the end-effector through the surface then the force clues will decrease the further the effector is moved inside. Thus, regardless of whether or not the operator chooses to comply with a fixture, its effects are felt only while in close proximity.

3.4.4 Multiple Fixtures for Box Interaction. In this example a *surface–surface fixture* (Fig. 7) is used to aid the operator in keeping the bottom cylindrical portion of the end-effector (actually the socket of an impact wrench) aligned approximately 1 cm below the top surface of the box at left of the picture. Moving in the plane of the surface requires no effort while moving either up or down requires a deliberate effort. This fixture provides an imaginary plane on which the operator may slide in order to contact the side of the box with the socket. Without the fixture it is difficult for operators to maintain just the right height—if too high then the socket will miss the box, while if too low the upper portion of the impact wrench will contact the box. Note that a second surface fixture has activated on the side of the box as the system anticipates the operator's next motion into contact.

3.4.5 Multiple Fixtures for Bolting/Unbolting. Here a number of fixtures are used to assist the operator in performing a bolting/unbolting task using an impact wrench. A *point-line fixture* (Fig. 8) has aided the operator in bringing the end-effector to a point where it is vertically over the bolt. The two square fixtures between the end-effector and the bolt form a virtual push-button. The upper surface (a *command surface–surface fixture*) acts as the top surface of the button while the lower one (a *repulsive surface–surface fixture*) acts as a detent limit. By pressing the impact wrench down through the upper surface the operator commands the system to activate the impact wrench.

All three of these fixtures are attached to the bolt hole so they will move with it if the world model is altered. They have activated in this case only because the end-effector is an impact wrench and it has a socket of the appropriate size and the bolt hole contains a bolt. They would not have activated otherwise. A surface fixture on the top face of the box has also been activated. This is because the system cannot be certain whether the operator intends to slide down the line fixture to activate the impact wrench or alternatively whether they intend to move away from the bolt and touch the surface with the empty socket. In cases such as this the system activates

several fixtures and allows the operator to choose, by their actions, those that are most appropriate.

4 Transforming Operator Interaction

The operator's interaction with the master arm is transformed based on the position of the virtual camera through which the operator chooses to view the simulated remote environment. The idea is to ensure that motion of the master arm feels very natural to the operator. For example, if the operator moves the master arm to the left then the simulated slave should move toward the left of the screen. If the operator now chooses to view the slave site from a different angle, a motion of the master arm to the left should still move the slave toward the left of the screen. It has been shown that aligning frames in this way can dramatically improve operator performance (Rasmussen, 1992).

Scaling is also accommodated so that the ratio between the distance with which the master arm is moved and the distance which the picture of the simulated slave moves on-screen remain approximately constant. The operator can "zoom in" to view small details and perform delicate motions, then "zoom back" to view the entire remote site, performing coarser actions. The force clues due to synthetic fixtures are scaled along with the screen image so that fixtures provide greater force clues when they appear close and lesser clues when distant. The use of appropriate synthetic fixtures can reduce the need to perform scaling since they assist in performing precise motions even while the operator is viewing the scene on a relatively coarse scale.

These transformations of the operator's interaction appear very natural. After moving the virtual camera novice operators are often quite unaware that this has caused the master arm motion to be transformed differently.

5 Command Generation

The operator's actions within the virtual environment are observed in order to generate, on-line and in real-time, robot instructions to cause the remote manipulator to perform the task. These commands are simple, performing a single action, and take relatively little time (typically around 500 msec) to execute. Typical commands are, in colloquial terms:

- Move forward 3 cm.
- Slide 2 cm to the right and expect to encounter a surface.
- Close the gripper and expect to encounter an object 3–4 cm wide.

Typically only one, or two commands are required for each second of task execution. These commands are thus at a relatively high level when compared with the joint servo information used in conventional teleoperation and are well suited to transmission over a narrow-bandwidth acoustic link.

The process of translating operator actions into discrete commands is complicated by ambiguous motions. For example, if the operator moved close to, but not exactly in contact with, a surface then the system must guess whether or not the operator really intended the slave to contact that surface. Such situations can be avoided by appropriate use of synthetic fixtures. The visual clues due to active fixtures give the operator some indication of what commanded actions the system is capable of performing while the force clues assist the operator in making very deliberate motions from which it is much easier to determine intent.

5.1 Generating Robust Commands

It is unrealistic to assume that the world model at the master station will ever be perfect and, even if it were perfect, it would be unreasonable to expect that the slave would always be able to execute each command perfectly. This is particularly true given the unpredictability of the subsea environment. In generating commands we aim to minimize cases where problems may occur.

The symbolic nature of the generated commands gives the system some natural tolerance for positioning errors. Performance in the presence of positional uncertainty is further improved by specifying motions as desired final positions relative to symbolically defined locations in the world. For example, consider the case where the operator moved the end-effector down onto a surface and

then moved up 2 cm. In this case the system would assign a symbolic label to the position contacting the surface and then the subsequent motion away from the plane would be specified relative to that symbolic position. As a result both the simulated end-effector at the master and the real end-effector at the slave should be positioned 2 cm above the surface despite the fact that the world model may be imperfect. If the operator now moved up another 2 cm the slave would be commanded to move to a position 4 cm above the symbolically labeled surface. Specifying motions in this way helps prevent positioning errors in consecutive motions from accumulating.

The system also takes an active approach to avoiding problems due to positional errors. We recognize that some motion commands are much more tolerant of modeling uncertainties than others and provide fixtures to assist the operator in performing these actions while dissuading them from performing less reliable motions.

The advantage of these techniques is that they allow the system to cope with modeling and positioning errors without requiring any changes to the world model at the master station. They are thus unaffected by the transmission time delay between master and slave sites. However, they will cope only with moderate modeling or positioning errors. Larger errors and other unexpected events are considered to be exceptional conditions.

6 Handling Exceptional Conditions

There are three stages to handling exceptional conditions, detection, diagnosis, and recovery. In the teleprogramming system we rely on autonomous detection, shared diagnosis, and manual recovery. We believe this is an appropriate use of technology. We rely on the operator handling the complex reasoning required to recover from errors while automating the tedious and time-consuming task of observing the slave's actions looking for errors.

Autonomous detection is desirable because it allows the operator to continue working after a command has been generated without needing to wait and observe the results of that command. Without autonomous detection the operator is forced to continuously observe the slave—this means he or she cannot work while the slave is working. To be effective autonomous detection has to be perceived as reliable by the operator. In the teleprogramming system exceptional conditions are detected at the slave site (Lindsay, 1992) by comparing actual sensory data with that predicted by the master station (and encoded symbolically within transmitted commands).

Error diagnosis is shared by the system and the operator. The system notifies the operator as to when the error was detected and how it was detected. It also provides some suggestion as to the cause of the error. This is implemented as a look-up table at the slave, for a particular error while in a particular behavioral state there is a hard-coded "best guess" as to what may have caused the error. The system does not make any attempt at performing complicated reasoning about what caused the error—that is the operator's job.

Recovering after an error is the operator's responsibility. He or she must evaluate the situation and select an appropriate response. This may entail modifying the positions of objects in the world or changing the recorded results of previous interactions with the environment.

7 Operator Interface for Error Recovery

When an error is detected at the remote site it notifies the master station. The master station then "winds back time" to the point where the error occurred. The effect for the operator is very much like watching a movie in reverse—all of the actions performed after the error command was transmitted are undone.

The operator is then apprised of the cause of the error by both a text message and, in some cases, a visual clue on the screen. An example text message is "Unexpected force detected in +Z dirn" and in this case an arrow is overlayed on the graphics indicating the +Z direction for the task frame in which the error occurred. The operator has the option, at this point, of stepping either backward or forward through the generated commands. For each command displays are available of both the commanded slave positions and the actual slave motion.

Since each transmitted command is only for a single short (typically around 500 msec) action then knowing which command generated the error message gives the operator some indication as to causality—there is only so much that can go wrong with each simple command. However, this information, coupled with the text message from the slave, is not usually sufficient for the operator to diagnose and correct the error. The reason is that this information indicates only what the slave "thought" happened. In actuality it may be that the error was erroneously generated and that nothing actually went wrong or, alternatively, it may be that in some previous command an error occurred but was not detected. As a result the operator must look not only at what may have caused the reported error but also at whether that error was itself erroneous. To do this the operator needs more information.

The comparison of actual and commanded positions just before the error occurred is perhaps the least-cost form of error analysis. Positions are sent continually as the slave is running. As a result, when an error message is received at the master station, it has already received the actual positions for commands preceding the error and this information is immediately available to the operator. This positional information is very useful in analyzing frequently occurring errors since the operator becomes accustomed to the characteristic "positional signature" for these errors. For example, in the case of lowering a bolt into a hole, if the commanded position would have put the bolt inside the hole, but the actual position shows only the end of the bolt on the surface, then it is likely that the slave missed the hole and is pushing the end of the bolt against the surrounding surface.

Pictures of the real remote site provide the operator with an excellent tool for diagnosing errors. Due to the limitations of an acoustic communications channel we cannot currently supply pictures for the operator at video rates. However, it is possible to continuously snap pictures at the remote site and then, when an error is detected, transmit images showing the state of the slave just before and/or just after the error was detected. Furthermore, because the model of the world includes some knowledge of the current task and context, we can predict in advance which direction the remote site cameras should be aimed in order to best aid in diagnosing any error. It should also be possible to predict the portions of the image that are most significant. This can then be used to aid compression of the image—transmitting high resolution information for those parts of the image and lower resolution information for other areas.

In recovering from an error the operator has the option of editing the world model to bring it into closer alignment with the real world. He or she also has the option of overriding the system's assumptions about the success or failure of previous commands. The operator can tell the system that any previous command failed (even though the system may have "thought" that it succeeded) or that any previous command succeeded (even though the system may "think" that it failed). This ability is important because the state of the world model influences the system's ability to aid the operator. For example, if the system thought the slave had unscrewed a bolt but the operator determined that to be false, then the operator must be able to go back to that command, correct the system's assumption, and then continue with the task. Since the system is now aware that the hole still contains a bolt it can again aid the operator in moving over that bolt.

The operator also has the ability to change the "conservativeness" of the generated commands. For example, when unscrewing a bolt it is possible to double-check that the bolt was correctly extracted by tapping it on a nearby surface. When a high level of conservativeness is chosen the system will automatically insert commands for "tapping" whenever an unscrew operation is performed. This costs bandwidth when the commands are transmitted and time when the commands are executed, but it saves time in those cases in which unbolting is unsuccessful by detecting them as soon as they occur. By allowing the operator control over this aspect of command generation we allow the intelligence and experience of the operator some influence on the form of the generated commands.

There is a trade-off in deciding how much, and in what form, information should be provided to the operator. If too little information is provided then the operator may incorrectly diagnose the exceptional condition. This wastes time since failing to correctly rectify

one problem invariably leads to others occurring later. Alternatively, if the operator is provided with too much information then bandwidth and time are wasted moving this data from the slave to the master station. It is expected that the optimal information display will vary depending on both the particular task being performed and the particular human operator performing that task. Much experience with real practical subsea applications will be necessary to determine suitable displays.

7.1 Coding around Errors

It is very tempting to avoid exceptional conditions by explicitly coding for them at the slave site. While this will certainly reduce or eliminate problems from that particular error, it is undesirable because it introduces a planning and programming phase that increases costs and makes immediate task execution impossible. It also raises command complexity. This increases command execution times, making it more difficult for the slave to reliably detect errors as well as making it more difficult for the operator to determine causality should an error be detected. By keeping commands simple we simplify the implementation of the slave without compromising the power of the system—complex actions may still be performed by encoding them as a sequence of simple commands for transmission to, and execution by the slave. This also gives the master system considerable control over how complex actions will be performed. Operator interaction can be allowed to influence the generated commands, allowing him or her to change the way complex actions are performed at the slave in response to current conditions.

8 Implementation

The software used to implement the master station is composed of a series of logical processes (see Fig. 9). The joint and Cartesian level processes run on JIFFE, a 40 MFlop VME-based co-processor (Andersson, 1989). All other processes run on a Silicon Graphics 4D/35 Elan machine. Update rates for the graphics display approach 30 Hz when doing nothing but update a single

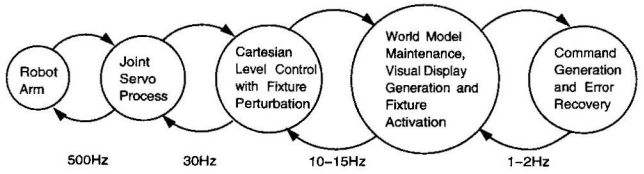

Figure 9. *Software implementation showing communication rates between logical processes.*

view of the remote site, but more realistic implementations, which display multiple views and handle command generation and error recovery, have update rates closer to 10 Hz.

For reasons of efficiency the calculations for implementing synthetic fixtures are split between the graphics workstation and the high-speed coprocessor controlling the master arm. All activation and filtering operations are performed on the graphics machine while the calculations needed for providing force clues are performed on the JIFFE coprocessor. All of the computation for a single fixture (for example the closest-surface–surface fixture) is performed in approximately 1 msec so it is quite feasible to have a number of simultaneously active fixtures and still achieve a reasonable Cartesian cycle update rate.

9 Conclusions

The teleprogramming paradigm provides a means to perform teleoperative tasks efficiently even when the communication between operator and remote sites occurs via a low bandwidth, high delay, communications link.

Synthetic fixtures have had a pervasive effect on the system. By providing force and visual clues they assist the operator in performing actions with speed and precision. By helping the operator to make deliberate, nonambiguous motions they simplify the task of translating action into robot commands. By guiding the operator's choice of actions they help him or her avoid the generation of commands that are likely to fail in the presence of positional uncertainty. A number of fixtures have been implemented. Examples have been presented showing

both simple single-fixture applications as well as complex multi-fixture systems.

Fixtures will aid the operator in commanding the remote robot with precision but, since they are generated using the world model, it is important to consider what happens when that model is inaccurate. Small modeling/positioning errors are accommodated by the slave robot's natural mechanical compliance. The system copes with moderate errors by specifying motions relative to symbolically labeled features. Larger modeling or positioning errors as well as other unexpected events are considered to be exceptional conditions.

The system employs technology appropriately. It automates the tedious and time-consuming task of detecting exceptional conditions while relying on a human operator to provide the reasoning and intelligence necessary to cope with the wide variety of unexpected events that occur in real-world interaction.

By keeping commands simple (a single action) and short (around 500 msec) we simplify the slave implementation, while making it easier for the operator to determine causality should an error occur. Complex actions are performed by transmitting a sequence of simple commands. This gives the master station some control over how complex actions should be performed and allows the operator to select an appropriate level of command conservativeness—choosing between slow and safe or fast and risky command sequences.

There are a number of areas in the system where compromise is necessary in order to meet the conflicting requirements of rapid task performance and low bandwidth communication. It is expected that optimal levels will vary based not only on the task but also on the preferences of each individual human operator. Only by applying this technology to real subsea manipulation tasks will good compromise points become clear. This is the subject of current research.

Acknowledgments

The authors wish to gratefully acknowledge the assistance of the Woods Hole Oceanographic Institution, whose collaboration is making possible the application of this work to real subsea manipulative tasks. This material is based upon 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. (1993). Teleoperation with virtual force feedback. *Third International Symposium on Robotics Research,* 125–131.

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

Autocad (1989). *AutoCAD Reference Manual.* Autodesk Inc.

Bejczy, A. K., Venema, S., & Kim, W. S. (1990). Role of computer graphics in space telerobotics: Preview and predictive displays. In *Cooperative intelligent robotics in space* (pp. 365–377). *Proceedings of the SPIE,* 1387.

Bier, E. A., & Stone, M. C. (1986). Snap-dragging. *SIGGRAPH Proceedings, 20*(4), 233–240.

Bruno, G., & Morgenthaler, M. K. (1991). Real time proximity cues for teleoperation using model based force reflection. In *Cooperative intelligent robotics in space II* (pp. 346–355). *SPIE,* 1612.

Dunbar, R. M., Klevelant, H., Lexander, J., Svensson, S., & Tennant, A. W. (1990). Autonomous underwater vehicle communications. In *ROV '90* (pp. 270–278). The Marine Technology Society.

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, The University of Pennsylvania.

Funda, J., Lindsay, T. S., & Paul, R. P. (1992). Teleprogramming: Toward delay-invariant remote manipulation. *Presence, 1*(1), 29–44.

Hirzinger, G. (1993). ROTEX the first robot in space. In *International Conference on Advanced Robotics,* 9–33.

Hollis, R., & Salcudean, S. E. (1993). Lorentz levitation technology: A new approach to fine motion robotics, teleoperation, haptic interfaces and vibration isolation. In *Preprints, Sixth International Symposium on Robotics Research.*

Kim, W. S., & Bejczy, A. K. (1991). Graphics displays for op-

erator aid in telemanipulation. In *IEEE International Conference on Systems Man and Cybernetics,* Charlottesville, VA.

Kim, W. S., & Stark, L. W. (1989). Cooperative control of visual displays for telemanipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1327–1332).

Kotoku, T., Tanie, K., & Fujikawa, A. (1990). Force-reflecting bilateral master-slave teleoperation system in virtual environment. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space,* 295–298.

Lindsay, T. S. (1992). *Teleprogramming: Remote site robot task execution.* Ph.D. thesis, The University of Pennsylvania.

Machida, K., Toda, Y., & Iwata, T. (1990). Graphic-simulator-augmented teleoperation system for space applications. *Journal of Spacecraft and Rockets, 27*(1), 64–69.

Millman, P., Stanley, M., Grafing, P., & Colgate, J. E. (1992). A system for the implementation and kinesthetic display of virtual environments. In *SPIE, 1833: Telemanipulator technology* (pp. 49–56). Boston, MA.

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

Oyama, E., Tsunemoto, N., Tachi, S., & Inoue, Y. (1993). Experimental study on remote manipulation using virtual reality. *Presence, 2*(2), 112–124.

Rasmussen, D. (1992). A natural visual interface for precision telerobot control. In *SPIE, 1833: Telemanipulator technology* (pp. 170–179). Boston, MA.

Rosenberg, L. B. (1992). *The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments.* Technical Report AL-TR-1992-XXXX, USAF Armstrong Laboratory, Wright-Patterson AFB OH (in press).

Sayers, C., & Paul, R. (1993). Synthetic fixturing. *Advances in robotics, mechatronics and haptic interfaces (DSC-Vol. 49), ASME Winter Annual Meeting, New Orleans,* 37–46.

Sayers, C., Paul, R., & Mintz, M. (1992). Operator interaction and teleprogramming for subsea manipulation. *Fourth IARP Workshop on Underwater Robotics, Genova, Italy.*

Sheridan, T. (1993). Space teleoperation through time delay: Review and prognosis. *IEEE Transactions on Robotics and Automation, 9*(5), 592–606.

Sutherland, I. E. (1963). Sketchpad—a man-machine graphical communication system. *Proceedings—Spring Joint Computer Conference (AFIPS),* 328–346.

Tanie, K., & Kotoku, T. (1993). Force display algorithms. *Lecture notes for workshop on force display in virtual environments and its application to robotic teleoperation* (pp. 60–78). IEEE International Conference on Robotics and Automation.

Unruh, S., Faddis, T., & Barr, B. (1992). Position assist shared control of a force reflecting telerobot. *SPIE, 1833: Telemanipulator technology* (pp. 113–121). Boston, MA.

Vertut, J., & Coiffet, P. (1986). *Robot Technology* (Vol. 3). Teleoperations and Robotics. Prentice-Hall.

Wanger, L. R., Ferwerda, J. A., & Greenberg, D. P. (1992). Perceiving spatial relations in computer-generated images. *IEEE Computer Graphics and Applications, 12*(2), 44–58.