February 1993

Model Based Teleoperation to Eliminate Feedback Delay NSF Grant BCS89-01352 - 3rd Report

Richard P. Paul
University of Pennsylvania

Thomas Lindsay
University of Pennsylvania

Craig Sayers
University of Pennsylvania

Matthew Stein
University of Pennsylvania

Desikachar Venkatesh
University of Pennsylvania

Model Based Teleoperation to Eliminate Feedback Delay NSF Grant BCS89-01352 - 3rd Report

Abstract

We are conducting research in the area of teleoperation with feedback delay. Significant delays occur when performing space teleoperation from the earth as well as in subsea teleoperation where the operator is typically on a surface vessel and communication is via acoustic links. These delays make teleoperation extremely difficult and lead to very low operator productivity. We have combined computer graphics with manipulator programming to provide a solution to the delay problem. A teleoperator master arm is interfaced to a graphical simulation of the remote environment. Synthetic fixtures are used to guide the operators motions and to provide kinesthetic feedback. The operator's actions are monitored and used to generate symbolic motion commands for transmission to, and execution by, the remote slave robot. While much of a task proceeds error free, when an error does occur, the slave system transmits data back to the master environment where the operator can then experience the motion of the slave manipulator in actual task execution. We have also provided for the use of tools such as an impact wrench and a winch at the slave site. In all cases the tools are unencumbered by sensors; the slave uses a compliant instrumented wrist to monitor tool operation in terms of resulting motions and reaction forces.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-93-40.

Model Based Teleoperation To Eliminate Feedback Delay NSF Grant BCS89-01352 Third Report

MS-CIS-93-40
GRASP LAB 345

Richard P. Paul
Thomas Lindsay
Craig Sayers
Matthew Stein
Desikachar Venkatesh

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

March 1993

Model Based Teleoperation to Eliminate Feedback Delay NSF Grant BCS89-01352 Third Report

Richard P. Paul
Thomas Lindsay Craig Sayers
Matthew Stein Desikachar Venkatesh

The University of Pennsylvania
GRASP Laboratory
Philadelphia PA 19104

February 1993

Abstract

We are conducting research in the area of teleoperation with feedback delay. Significant delays occur when performing space teleoperation from the earth as well as in subsea teleoperation where the operator is typically on a surface vessel and communication is via acoustic links. These delays make teleoperation extremely difficult and lead to very low operator productivity.

We have combined computer graphics with manipulator programming to provide a solution to the delay problem. A teleoperator master arm is interfaced to a graphical simulation of the remote environment. Synthetic fixtures are used to guide the operators motions and to provide kinesthetic feedback. The operator's actions are monitored and used to generate symbolic motion commands for transmission to, and execution by, the remote slave robot. While much of a task proceeds error free, when an error does occur, the slave system transmits data back to the master environment where the operator can then experience the motion of the slave manipulator in actual task execution. We have also provided for the use of tools such as an impact wrench and a winch at the slave site. In all cases the tools are unencumbered by sensors; the slave uses a compliant instrumented wrist to monitor tool operation in terms of resulting motions and reaction forces.

Contents

1 Introduction **1**

2 Past Research **3**

3 Current Research **4**

3.1 Remote Site Robot Task Interaction 4

3.2 Master Station, Interacting with Uncertainty 5

3.3 Detecting Contact 6

3.4 Error State Display and Kinesthetic Feedback 7

3.5 Behavioral Based Controller 7

4 Documentation **8**

A APPENDICES **9**

A.1 Teleprogramming: Remote Site Robot Task Execution 9

A.2 Improved Instrumented Compliant Wrist Design 117

A.3 Operator Interaction and Teleprogramming for Subsea Manipulation 153

1 Introduction

We are conducting research in the area of teleoperation with feedback delay. Significant delays occur when performing space teleoperation from the earth as well as in subsea teleoperation where the operator is typically on a surface vessel and communication is via acoustic links. These delays make teleoperation extremely difficult and lead to very low operator productivity. We have developed a novel technique, combining virtual reality and manipulator programming, to solve the delay problem by interfacing a teleoperator master arm to a graphical simulation of the remote environment. The operator works, without delay, in the virtual world while his or her actions are monitored to provide both kinesthetic and visual feedback. Based on these motions the system generates symbolic robot program commands for transmission to the remote slave. The slave robot executes these symbolic commands delayed in time.

Based on a model of the remote site we create a virtual reality for the operator. The operator interacts with the model using a commercial PUMA 250 robot manipulator which serves as a six-degree-of-freedom input device. By moving this master arm the operator can control the displayed image of both the remote slave arm and any object that it is carrying. We monitor the position of the slave arm, in the geometric model, to detect the proximity of work objects. This proximity information is combined with task knowledge to selectively activate synthetic fixtures which provide task-dependent kinesthetic feedback to the operator. For example if a face on the end effector were brought close to a face on an object and if it were appropriate for those two faces to be in contact then the system would activate a surface synthetic fixture to assist the operator in reaching and then maintaining a face-face contact.

With these capabilities, the operator can not only see what is going on but can also *feel*, kinesthetically, the objects represented in the display. When the inside of a box is displayed the operator can feel along a surface to a corner between two surfaces; the operator can slide along the edge into the corner of the box. The combination of the visual display with kinesthetic feedback from the scene provides an extremely strong telepresence. The operator can really feel that she/he is "there."

The automatically generated robot commands for execution at the remote site are very simple consisting primarily of free space moves and guarded motions. Whenever the operator brings two objects together in the virtual world a guarded move is sent to the remote site. We do not send any conditionals such as "if a happened then do b else do c," as if an error occurs, we can wait for the operator to interpret the situation and to generate the appropriate corrective actions. Conditionals have plagued robot programming in the past as every situation must be anticipated, and every possible outcome of an action predicted and pre-programmed. As any robot programmer knows, it is impossible to account for everything that can happen during task execution, especially when one realizes that the corrective action for every error will itself involve errors – a hopeless situation.

The symbolic robot commands are executed by a slave manipulator at the remote site. Notice that the time delay between the operator input and the slave execution may be quite arbitrary; the slave is simply following along as if someone were sitting at a terminal writing a program and executing it line by line. Of course, something might not work exactly in the slave's world as it did in the image world. At this point we would alert the operator by "flashing" the image and then returning the display to the point at which the slave is "hung-up." We also change the constraints on the input device to correspond to the situation the slave has detected. The operator has the option of "feeling" the motion of the slave leading up to the error based on information sent back from the slave to the master station when the error was detected. This information is transmitted from the slave to the master station using an identical instruction format to that which the master normally uses to instruct the slave. The operator then resumes task execution from this new state. Once again, these actions would be translated into symbolic robot commands and sent to the slave.

2 Past Research

During the second year of the grant the master station was completed, to the extent that the operator could perform a task in the modeled world and automatically generate robot manipulator instructions for the slave. The slave station was also completed, to the extent that robot instructions could be received and then executed after introducing an appropriate delay. A low level language was developed and a parser written for the slave manipulator. Delays in parsing instructions by the slave were solved by a double buffering scheme so that the slave manipulator was kept in synchronization with the master but with a constant delay. The task we choose was the exploration of a box with the operator finding the box, its sides, bottom and corners, etc. The slave was delayed by about 3 seconds. Only primitive error recovery was developed, but, by the end of the year the system was quite reliable and we could sustain operation for up to 30 minutes at a time.

3 Current Research

3.1 Remote Site Robot Task Interaction

Execution at the remote site is in a semi-autonomous mode. A slave robot is programmed to perform the task step-by-step. The slave robot has enough task information to act autonomously for the duration of the round-trip communication delay. Low-level manipulator control is based on the hybrid force/position control method and makes use of an instrumented compliant wrist (see Appendix A.2). The remote slave system is described fully in [2] (see Appendix A.1).

Commands sent from the master-station do not specify force levels or take into account, in any detailed way, the dynamics of the slave system. It is up to the slave system to set force limits and to compensate appropriately for the dynamics and frictional effects of task execution. Contact state changes are based on running averages and statistical methods. In this way we are able to work on objects which may be soft or hard, smooth or rough.

The remote system also provides for tool usage. An impact wrench and winch are available for its use. The impact wrench is used to insert and remove nuts and bolts. Its actions are monitored using both the time history of the displacement of the manipulator holding the wrench and the reaction forces experienced at the compliant instrumented wrist. In this way it is possible to use tools without needing to have them instrumented or to provide an instrumentation interface.

The winch is used to overcome large gravity forces with the manipulator simply guiding the horizontal motion of an object. The winch provides far greater lift capability than the manipulator would be capable of exerting. Once again, this tool is uninstrumented. The motion of the load during winching operations is detected using the resultant motion of the manipulator, which is also holding onto the load and complying with vertical axis forces.

3.2 Master Station, Interacting with Uncertainty

Improvements to the master station for teleoperation systems have generally focussed on improving operator performance by providing more sophisticated master arms or improved visual displays (see Appendix A.3). We proposed synthetic fixturing [6], where the master system actively guides the operator's motions in one or more degrees of freedom, as a means of increasing precision and speed without the need for sophisticated and expensive hardware. Fixturing is accomplished by giving the manipulator a tendency to drift, in one or more degrees of freedom, toward predetermined task-dependent positions. The force is sufficiently large that an operator who wishes to make use of the fixture can just relax and let it pull their hand along and yet sufficiently small that an operator who wishes to move in a different direction can still get there – he or she just needs to push a little harder.

Fixtures increase precision by reducing the positional uncertainty associated with the operator's motions in the fixtured degrees of freedom. They also decrease the accuracy to which the operator must move and hence increase the speed with which they can perform teleprogramming tasks. As a result fixtures avoid the need to trade speed for accuracy.

Synthetic fixturing has the additional, more subtle, benefit of reducing the uncertainty associated with the command generation process. As the system observes the operator's actions it must interpret the operator's input and transform that into a command sequence for execution at the remote site. Now, if the operator complies with, for example, a line fixture then motion of the master will be along a straight line and its obvious to the system that this motion was what the operator desired. Any deviation from the fixture line requires that the operator overcome a distinct resistance which is very apparent to the system. Fixturing thus has the effect of reducing both the operator's uncertainty as to which commands will be generated and the system's uncertainty as to which commands the operator would wish created.

While fixtures will help the operator perform an action they won't assist the operator in deciding which action to perform. Aiding the operator in such a decision requires that they have some information regarding the likelihood of any given action being correctly executed in the uncertain world at the slave site. To this end it has been proposed [5] that color clues be employed to provide the operator with a visual measure of uncertainty. It is suggested that the provision of information about both the position and positional uncertainty of objects should enable users to compromise between the fast, but risky, approach of directly manipulating objects whose position is uncertain and the slow, but safe, method of feeling out the position of each object before attempting to work with it.

3.3 Detecting Contact

In teleoperation the reliable detection of contact between the slave end effector and objects in the world is vital. It is also important to maintain required contact forces between the end effector and the environment. The detection of a contact or collision is accomplished using a time series data acquired from potentiometers mounted on the instrumented wrist. An attempt is being made to detect these contacts or collisions in the frequency domain. The wrist is modeled as a system of linear and torsional springs. The solution of the system equations provides the frequencies of the multiple modes of vibration when the wrist undergoes a state change as in contact or collision. The experiments were performed using a PUMA-560 slave arm equipped with a wrist sensor to measure the forces and torques. An accelerometer is also fitted on this sensor to measure the acceleration of the wrist. The acceleration data is acquired at a sampling frequency of 1000 Hz. The analysis of this data is carried out on a 1024 point Fourier transform, resulting in a frequency resolution of approximately 1 Hz. The frequency spectrum indicated peaks at two different frequencies corresponding to two modes of vibration, the values being in the vicinity of that obtained through the mathematical model. There is a change only in the amplitude of the frequency spectrum and no appreciable change in the frequency values are observed. The discrimination of the change of end effector states based only on the amplitude of the spectrum may not be a reliable measure. The dynamic model of the wrist developed in this analysis could be used to improve upon the control strategies of the slave arm.

3.4 Error State Display and Kinesthetic Feedback

Research at the master station also involved the development of schemes for error diagnosis and recovery. The symbolic command language for communication from the master to the slave was extended and utilized for communication in the opposite direction. The slave sends reply statements to the master station indicating motions performed and forces encountered. This information is conveyed to the operator by a technique which allows the operator to re-experience the kinesthetic information obtained at the slave site. With this improved feedback information, error diagnosis and correction is more easily performed.

3.5 Behavioral Based Controller

While our work is directed to undersea operation similar time-delayed teleoperation problems occur in space. We have considered the task of slicing a thermal blanket. This task requires a higher degree of autonomy from the slave as it must now react in a more complex manner to the environment. A behavioral based controller has been implemented at the slave site to provide that degree of autonomy. The behavior based controller is activated by a command from the master. When active the controller operates independently, while returning reply messages to the master. When the behavior based controller encounters a conflict in behaviors this indicates a situation which controller is incapable of handling. In this situation the behavior based controller seeks operator assistance. Using kinesthetic replay, the operator may diagnose the situation and provide corrective motions to the slave. Utilizing this form of control, the thermal blanket slicing task was performed.

4 Documentation

[1] Thomas Lindsay and Richard P. Paul. Improved instrumented compliant. Technical Report MS-CIS-92-77, University of Pennsylvania, 1992.

[2] Thomas S. Lindsay. Teleprogramming: Remote site robot task execution. Technical Report MS-CIS-92-64, University of Pennsylvania, 1992.

[3] Richard Paul, Thomas Lindsay, and Craig Sayers. Time delay insensitive teleoperation. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 247–254, July 1992.

[4] Richard Paul, Thomas Lindsay, Craig Sayers, and Matthew Stein. Time-delay insensitive virtual-force reflecting teleoperation. In *Artificial Intelligence, Robotics and Automation in Space*, pages 55–67, September 1992. Toulouse, France.

[5] Craig Sayers, Richard Paul, and Max Mintz. Interacting with uncertainty. In *Telemanipulator Technology.* SPIE Proceedings Vol.1833, November 1992.

[6] Craig Sayers, Richard Paul, and Max Mintz. Operator interaction and teleprogramming for subsea manipulation. In *Fourth IARP Workshop on Underwater Vehicles, Genova, Italy*, November 1992.

[7] Craig Sayers, Richard Paul, and Max Mintz. Operator interaction and teleprogramming for subsea manipulation. Technical Report MS-CIS-93-15, University of Pennsylvania, 1993.

[8] Matthew R. Stein and Richard P. Paul. Kinesthetic replay for error diagnosis in time delayed teleoperation. In *SPIE OE/Technology '92: Telemanipulator Technology*, volume 1833, 1992.

A APPENDICES

A.1 Teleprogramming: Remote Site Robot Task Execution

Teleprogramming: Remote Site Robot Task Execution

Thomas S. Lindsay

A Dissertation

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1992

Richard P. Paul
Supervisor of Dissertation

Portonovo S. Ayyaswamy
Graduate Group Chairperson

Acknowledgments

I would like first to thank Dr. Richard Paul, whose inspiration and support made this dissertation possible. His insight and his humor have make the experience fulfilling and enjoyable. It seems hard to believe that only four years have passed since he first suggested I acquire an impact wrench!

I would also like to thank Dr. Ruzena Bajcsy for providing a finer collection of equipment and colleagues than anyone could hope for. A few of these colleagues need to be mentioned here for their special contributions to this work: Yangsheng Xu (for introducing me to instrumented compliant wrists); Pramath Sinha, Mario Campos, and Marcos Salganicoff (for their contributions to the wrist hardware and software); Janez Funda, Craig Sayers, and Matthew Stein (for their contributions to the teleprogramming project); Eric Paljug (for his expertise and patience with hardware and control questions); and the rest of the Graspees and staff members who have provided technical as well as moral support throughout my years here.

Thanks are also due for the two committee members not yet mentioned, Dr. Vijay Kumar and Dr. Burton Paul, for taking time to review my work. Many of their comments and suggestions have enriched this dissertation.

Finally, a special thank you to my wife Marianna for supporting me through these years, and for love, comfort, and an occasional prod at the right moments.

This work was supported by the National Science Foundation under Grant No. BCS-89-01352, "Model-Based Teleoperation in the Presence of Delay." Any opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the National Science Foundation.

Abstract

Teleprogramming: Remote Site Robot Task Execution

Thomas S. Lindsay

Supervised by Dr. Richard P. Paul

This dissertation describes a remote site robot workcell for teleoperation with communication delays on the order of 20 seconds. In these situations, direct teleoperation becomes impossible due to the delays in visual and force feedback. *Teleprogramming* has been developed in order to overcome this problem.

An integral part of teleprogramming is a semi-autonomous remote site robot system. The remote system is composed of a robot manipulator, sensors, controlling computer, and manipulator tools. The constraints on the remote site system and the amount of autonomy needed are defined partially by the teleprogramming system, and partially by the needs of the remote system. Development of the remote site system for teleprogramming evokes some pertinent research issues: low level manipulator control, semi-autonomous command execution, and remote site tool usage.

Low level manipulator control is based on a hybrid control scheme using wrist-based sensory feedback. Implementation of this control is presented and problems related to controlling the manipulator in an arbitrary frame are investigated.

High level commands are executed at the remote site in small autonomous steps. Implementation of tolerance checks and guarded moves are presented, including error detection and the detection of motion termination conditions in a partially known environment.

Power tools introduce redundant degrees of freedom into the manipulator/tool system. To control this redundant system, the tool is actively controlled in its natural degree of freedom and the corresponding degree of freedom in the manipulator becomes passive. Feedback for the manipulator/tool system is supplied by the wrist-based sensor. Two examples of sensing and control for tools are presented.

This research has resulted in the development of a remote site system for teleprogramming. The remote system, however, is both time-delay and input independent. The characteristics of the system, including the compliance, flexibility, and semi-autonomity, are useful to a wide variety of robotics applications, including manufacturing and direct teleoperation.

Contents

Acknowledgments iii

Abstract iv

1 Introduction 1

1.1 Problem Statement 3

1.2 Terminology and Definitions 5

1.3 Outline of Dissertation 6

2 Related Work 7

2.1 Modern Telerobotic Systems 7

2.1.1 Direct Teleoperation 8

2.1.2 Time Delayed Teleoperation 8

2.2 Teleprogramming 10

2.3 Limitations 13

2.4 Summary 13

3 The Remote System 14

3.1 Preliminary Experimental Results 16

3.2 Remote Site Research Topics 17

3.3 Summary 17

4 Hybrid Control Using an Instrumented Compliant Wrist 19

4.1 Background 20

4.2 Hardware 21
4.3 Hybrid Control 22
4.4 Remote Frames 27
4.5 Simulation of 3-DOF Arm 32
4.6 Experimental Results 39
4.7 Conclusions 41

5 Parsing and Execution of Semi-Autonomous Commands 43

5.1 Background 45
5.2 Command Language 46
5.2.1 Command Types 46
5.2.2 Sample Execution Environment 47
5.2.3 Real World Parameters 48
5.2.4 Example Program 49
5.3 Tolerance 51
5.3.1 Interpreting the Motion Commands 52
5.4 Collision Detection 55
5.5 Experimental Results 57
5.5.1 Data Distribution 57
5.5.2 Example Program 61
5.6 Limitations 65
5.7 Conclusions 66

6 Tool Usage 68

6.1 Background 69
6.2 Tool Control 69
6.3 Tool Sensing 71
6.4 Impact Wrench 72
6.4.1 Sample Program Utilizing the Impact Wrench 73
6.4.2 Experimental Results 75
6.5 Winch 76

6.5.1 Sample Program Utilizing the Winch 79
6.5.2 Experimental Results 82
6.6 Conclusions 86

7 Conclusion **88**
7.1 Contributions 89
7.2 Future Work 90

Bibliography **93**

List of Figures

2.1 Basic Teleprogramming Schematic 11

2.2 The experimental teleprogramming testbed 12

3.1 Preliminary Experimental Remote Site Environment 16

4.1 Instrumented Compliant Wrist 21

4.2 Conceptual Organization of Hybrid Controller 23

4.3 Remote Task Frames 28

4.4 Rotations about frames C and TF 29

4.5 Three degree of freedom robot with long final link 30

4.6 Simplified Control 30

4.7 Noise Input 32

4.8 Cartesian Position, $l_3 = 0$ 33

4.9 Joint Angles, $l_3 = 0$ 34

4.10 Joint Angles, Off-Diagonal Term = 110 34

4.11 Joint Angles, $l_3 = 120$ 35

4.12 Joint Angles, $l_3 = 120$, Reduced Gains 35

4.13 Root locus for 3-link manipulator 38

4.14 Close-up of root locus for 3-link manipulator 39

4.15 Cartesian Deflection, Off-Diagonal Term $= 30,000$ 40

4.16 Cartesian Deflection, $l = 30m$ 40

4.17 Cartesian Deflection, $l = 30m$, Reduced Gains 41

5.1 Generated Move for Guarded Motion 52

5.2 Executed Move for Guarded Motion 53

5.3 Force Detection Algorithm 56
5.4 Sensor Data from Stationary Manipulator 58
5.5 Sensor Data from Free Space Motion 58
5.6 Sensor Data from Sliding Motion 59
5.7 Free Space Sensor Data 60
5.8 One Contact Sensor Data 61
5.9 Illustrative Experiment 62
5.10 Force Data 63
5.11 Free Space - One Contact Collision 64
5.12 One Contact - Two Contact Collision 65
5.13 Undetectable Error Conditions 67
6.1 Wrist Outfitted With Impact Wrench 72
6.2 Impact Wrench Removing Bolt 75
6.3 Velocity of Robot While Removing Bolt 77
6.4 System Controlled Winch 78
6.5 Winch Experiment 81
6.6 Winch Lifting Box Top 83
6.7 Hook Velocity Data for Guarded Move (a) to (b) 84
6.8 Hook Force Data for Guarded Move (c) to (d) 85

Chapter 1

Introduction

Although robotics has its roots in teleoperation, it is only recently that major efforts have been made to use the technology of modern robotics in teleoperative systems. Major research efforts attempt (a) to increase human operator productivity, (b) to increase the level of autonomy of the remote system (in order to interact more intelligently with an unknown environment, and again to increase operator productivity), and (c) to overcome problems related to time-delayed teleoperation. Teleprogramming (which will be described herein) was designed for time-delayed teleoperation, yet the implementation touches on all of these areas.

The first automatic electric-powered teleoperator was developed in the 1940s to manipulate radioactive material [Goertz 1963]. Teleoperative systems are still used today for tasks in hazardous environments. As technology improves, the applications for teleoperative systems increase. For certain environments, such as shallow space and subsea applications (using unmanned, untethered submersibles), significant communication delays occur between the master and remote sites.

Delays in feedback make direct teleoperation impossible. Delayed visual feedback leads operators to adopt a move-and-wait strategy, which considerably increases the time to perform teleoperation tasks. Delayed force feedback can cause the system to become unstable, or, if the system is designed to remain stable, the performance will still degrade with longer delays[1]. Teleprogramming has been designed to eliminate these problems.

Teleprogramming bypasses the problems associated with delayed feedback from the remote site. Control loops are closed locally at both master and remote sites; the human operator receives direct visual and kinesthetic feedback based on interactions with a local task model of the remote environment: a virtual remote site. Based on operator interaction with the local model, commands are automatically generated and sent to the remote site. The remote manipulator executes these commands, compensating in real time for inaccuracies in position and force sensed locally.

For a successful time-delayed teleoperative system, some remote site autonomy is needed. The level of autonomy is dependent upon the time delay: when no delay exists, direct feedback is possible and no remote site autonomy is needed; when the communication delays far exceed the task performance time, the remote site needs to be completely autonomous. Although research in completely autonomous systems is progressing, state of the art autonomous systems cannot as yet interact intelligently to completely unexpected situations, which occur often in unstructured environments. In most situations, a human supervisor is at least desired, if not necessary. The current implementation of teleprogramming is designed for delays on the order of seconds. The level of remote site autonomy needed with this level of delay is manageable, and the human operator maintains an active role in the teleoperation control loop.

The remote site system must overcome the human operator's lack of remote site information. The operator lacks important environmental parameters, has poor abilities for imaging and modeling the remote site, and has no ability to directly verify actions. The remote system autonomy must therefore be able to compensate for local environmental parameters, must be able to execute commands that are inaccurate due to poor modeling, and must be able to verify proper execution of commands. The remote system designed for teleprogramming is compliant, semi-autonomous, and able to work in partially known, unstructured environments.

[1]Similar problems occur when communications between master and remote site are bandwidth-limited. This problem can occur when communicating to a remote vehicle via RF links.

The main application of the teleprogramming project in the GRASP Lab[2] is underwater exploration, intervention, and salvage, using unmanned, untethered submersibles. Currently, remotely operated tethered submersibles are used for operations that are too deep for divers. They are expensive to maintain and operate because the submersible must be supported by a surface ship. Untethered submersibles are being used for subsea survey operations at various depths, but are incapable of performing manipulation tasks. Unmanned, untethered submersible vehicles which can communicate to the surface via an acoustic link, have none of these problems. Multiple submersibles can operate in the same area, communicating with the surface via an acoustic/radio frequency buoy. Such systems could be easily deployed from a helicopter. Based on the speed of sound in water, however, acoustic communication links introduce significant communication delays between the master and remote sites.

Shallow space presents another application for teleprogramming. The cost of human activities in space, especially extra vehicular activity (EVA), are extremely high. Long EVA maneuvers are physically exhausting to astronauts, which can lower their productivity and safety. Teleoperative systems could eliminate the need for most EVA activities. However, delays in communication between earth and shallow space can easily be as much as 18 seconds, considering earth and satellite based relay stations, precluding direct teleoperation. Teleprogramming systems in space could be used for satellite repair and maintenance, space station maintenance, and other space activities, greatly reducing the need for human intervention in space.

1.1 Problem Statement

The teleprogramming system is a new approach for performing time delayed teleoperation. An integral part of the teleprogramming system is a semi-autonomous remote site (slave) system. Simplicity of the teleprogramming language (commands that are sent from the master site to the remote site) is necessary for the master system, as

[2]General Robotics and Active Sensory Perception Laboratory, University of Pennsylvania Dept. of Computer and Information Science, Philadelphia, PA. Ruzena Bajcsy, Director.

commands must be automatically generated by the system. However, the remote system must have the knowledge to apply these simple commands to the complex environment it must work in. Certain parameters of the remote site environment may not be available to the master system, such as surface friction and object masses, yet the remote system must compensate for these variables. Additionally, errors in the master site model of the remote environment force the remote system to work with inaccurate information, such as distances that are accurate only to a pre-defined tolerance. Finally, the remote system must autonomously determine the success or failure of a given command. Development of the remote site system for teleprogramming evokes some pertinent research topics, outlined below.

1. Development of a hybrid controller that acts semi-autonomously and allows the remote manipulator to interact with an unknown or partially known environment. Specification of motions, forces, and hybrid modes may be given in an task arbitrary frame. The limitations on the arbitrary task frame which exist when using wrist-based sensor feedback must be determined.

2. Command execution with the inclusion of dynamic (real-world) parameters. If the remote system cannot determine exact parameters, it must at least be able to compensate for sensor signals that occur as a result of the unknown parameters. Motion control with inaccurate distances, and identification of motion terminating conditions (collisions, loss of surface contact, etc.) in a noisy environment are two areas for research.

3. Manipulator tool usage, specifically tools that add a degree of freedom to the system. The manipulator/tool system thus has a redundant degree of freedom, and a method for control and sensing is needed. Further, the operator's task should not be further complicated by tool usage.

The major goal of this research is the development of a working remote site system for teleprogramming. The behavior of this system can then be studied, which may suggest improvements to both the remote system and the master system. With the completion of this work, a testbed for further research in teleprogramming is available.

Other goals include improvement to the understanding of wrist-sensor based hybrid control, advancement of semi-autonomous robot control, and advances toward a general tool usage control strategy for telerobotics.

Because the system is independent of time delays, the semi-autonomous control structure developed for the remote site of the teleprogramming system may have applications in non-time-delayed environments. It can be used as a flexible controller for direct robot programming, teleoperation without time delays, and replaying stored sequences of commands for manufacturing or other repetitive tasks. It can accept input from any other interface, such as an autonomous agent, a multi-agent controller, etc. The current implementation has been used for teleprogramming as well as other research projects in the GRASP Lab which utilize the remote site control structure for non-delayed applications.

1.2 Terminology and Definitions

The terminology of teleoperation developed when teleoperation first became a technology. The terms *master* and *slave* are traditional and commonly used, mainly because they are very descriptive of the teleoperation process. However, the term slave is condescending, especially to the research presented here! *Local site* and *remote site* are often used, respectively, in place of master and slave. However, in terms of a relative coordinate system, the "local" system for this research is actually the remote site! In order to reduce confusion (mainly the author's), the term *master site* is used for the system that the human interacts with as the teleoperation input device. *Remote site* is the term used for the system that interacts with the "remote" environment.

Other terminology in this paper may be confusing, and many terms have synonyms that are widely used. Some of the major terms used in this dissertation are presented in table 1.1, along with synonyms commonly found in related literature.

Terminology	Synonyms	Brief Definition
Contact State		The number and types (face-face, edge-face, etc.) of contacts between the manipulator and environment
Master Site	Local site operator's station	Site including the master manipulator and controllers where the human operator works
Remote Site	Slave site remote workcell distant site	Remotely located manipulator, sensors, controller, and tools at the site where work is to be done
Remote Task Frame		A task control frame that is not located at the wrist sensor
Task Frame	Task Coordinates Control Frame	Coordinate frame in which motions, modes, etc. are defined
Time Delayed Environment	Remote Environment	Environment located remotely from the master site. Communication time between master and this environment is significant
Hybrid Control	Hybrid Position/ Force Control	Control algorithm in which cartesian directions are independently chosen to be controlled by either position or force feedback

Table 1.1: Terminology and Definitions

1.3 Outline of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 reviews the current state of research in telerobotics and time-delayed teleoperation. Included in this chapter is an overview of the teleprogramming paradigm. Chapter 3 presents the details of the current remote site robot system for teleprogramming. Chapter 4 describes the hybrid control scheme used by the remote site robot. Command execution including command parsing is studied in chapter 5. Tool usage by the remote manipulator is discussed in chapter 6. Finally, conclusions, contributions, and future research areas are discussed in chapter 7.

Chapter 2

Related Work

The future of remote manipulation was once thought to be autonomous robot manipulator systems. However, advances in autonomous systems have not kept pace with the requirements of modern applications. With advances in the field of modern robotics, telerobotics has become a viable technology to replace human operators in many hazardous environments, including environments that are remote enough as to cause communication delays for master/remote site communications.

In this chapter, an overview of the modern teleoperative systems is presented, as well as a description of the teleprogramming paradigm as it is currently implemented.

2.1 Modern Telerobotic Systems

The technologies of modern robotics, including aspects of artificial intelligence, virtual reality, and computationally intensive simulations have recently been applied to teleoperation applications. Two types of telerobotic systems are under development: direct teleoperation systems using state-of-the art robotic techniques, and teleoperation systems which operate in time-delayed environments. Unfortunately, there is very little overlap between these two types of system: technology from one area is not usually applicable to the other. A brief mention will be made of direct teleoperation systems, and a more in-depth overview of time-delayed telerobotics will be presented.

2.1.1 Direct Teleoperation

Modern direct teleoperation systems utilize the latest advances developed for robotics and robotics-related technology. Two major examples are mentioned below.

One of the foremost applications of artificial intelligence (AI) in telerobotics is at the Electrotechnical Laboratory (ETL) in Japan, where it is used to increase human operator productivity, and simplify the human operator's task [Hirai et al. 1990]. ETL uses a combination of video and computers to model the environment, overlaying CAD models on a video view of the remote environment to enhance the model [Ogasawara et al. 1991].

NASA Ames Research Center (Mountain View, CA) has recently announced a research project for teleoperation on Mars using virtual reality (VR) techniques. How this system will react with time delays has not yet been addressed, but the system architecture has been successfully demonstrated on an unmanned submersible [Wav 1992].

2.1.2 Time Delayed Teleoperation

Time delayed teleoperation has been a major research topic for many years. Ferrell found that given a time-delayed teleoperation task with only visual feedback, a human operator tends to adopt a *move-and-wait* strategy [Ferrell 1965]. For a given length of task and time delay, this may not be practical. In response to this problem, Ferrell and Sheridan formalized supervisory control [Ferrell and Sheridan 1967], a broad strategy for balancing the workload between the human operator and a semi-autonomous remote system. Supervisory control addresses most of the issues involved with time-delayed teleoperation. However, as formalized by Ferrell and Sheridan it has some disadvantages. First, there is still the need for the human operator to acknowledge the completion of a command. As stated, "In the region of combined man-and-machine control, the operator is able to extend his open-loop 'moves' so that he gives fewer but more comprehensive commands. With fewer commands there are correspondingly fewer waits for correct feedback." These waits will inevitably add to task completion time. Second, branching is heavily utilized in the control of the remote site system. Branching for error recovery creates a large, tedious, and at times error-prone programming task for the human operator. Finally, the intercommunication needed to recover from an error need not be as intensive as Ferrell and Sheridan imply.

Niemeyer and Slotine have shown, using a two-ported model, that direct force feedback is possible and stable with delayed feedback loops [Niemeyer and Slotine 1991]. However, it is noted that performance still degrades as the time delay increases. Furthermore, the human operator and the environment are not included in the formulation. In order for the system to work in a manner that the human operator can understand, prediction algorithms are needed. Anderson and Spong have shown, using network theory, that force reflecting teleoperation is asymptotically stable for any time delay [Anderson and Spong 1992]. This research included models of the human operator and environment, but fails also to address the issue of system performance with delayed force feedback. The usefulness of force feedback after about one second of delay is questionable.

Kinematic and dynamic prediction algorithms are used extensively by Hirzinger et. al. in the ROTEX system[Hirzinger et al. 1989]. However, in this approach the remote site system and the delay lines must be accurately modeled, and the method is computationally intensive. This approach is unsuitable for unknown, unstructured environments.

NASA's Jet Propulsion Laboratory (JPL, Pasadena, CA) has been working for several years developing systems for satellite maintenance and repair in space [Bejczy et al. 1990]. Two robot arms are controlled by 6-degree of freedom (DOF) force-reflecting master devices. Certain repetitive remote site tasks have been preprogrammed, to increase productivity. In a delayed environment, the operator can see a "ghost manipulator" that moves in real-time, displaying the motion that the remote manipulator will follow in response to inputs from the human operator. This is only useful for free-space motions, as force feedback is not provided in the delayed system. Motion in contact with the environment is facilitated by shared compliance control. Motions which require force feedback are still executed using the move-and-wait strategy.

Recently Paul, Funda, et. al. have developed teleprogramming, described below, which is used for time-delay-invariant teleoperation. Performance of the system remains constant with any time delay. More information about teleprogramming can be found in [Paul et al. 1992, Funda et al. 1992, Funda 1991, Paul et al. 1990].

2.2 Teleprogramming

The teleprogramming concept bypasses the problems associated with delayed feedback from the remote site. Control loops are closed locally at both locations; the human operator receives direct visual and kinesthetic feedback from a local model of the remote site, and the remote manipulator receives direct sensory feedback from local sensors and autonomously compensates for inaccuracies in position and force. The operator interacts with the graphical model as if it were a direct teleoperation task in a virtual environment. The master system automatically generates a set of commands based on the interaction between the operator and the graphical model that are sent via the delayed communication link to the remote site. The remote site executes these commands, and either acknowledges successful execution or reports errors back to the master. Error recovery remains the responsibility of the human operator.

Using teleprogramming, teleoperation becomes delay-invariant: the amount of communication delay does not affect task performance. Although teleprogramming has been developed for time delays of one to 20 seconds, it may have applications with time delays of less than one second and greater than 20 seconds.

A teleprogramming testbed is operational in the GRASP lab. MERIONETTE (Model-based EditoR for Interactive ON-linE Teleprogramming in Time-delayed Environments) has demonstrated that teleprogramming is a feasible paradigm.

Any teleoperation system can be visualized as the interconnection of four elements: the human operator, the master system, the remote system, and the environment. The master and remote systems are connected by a communication link. Feedback

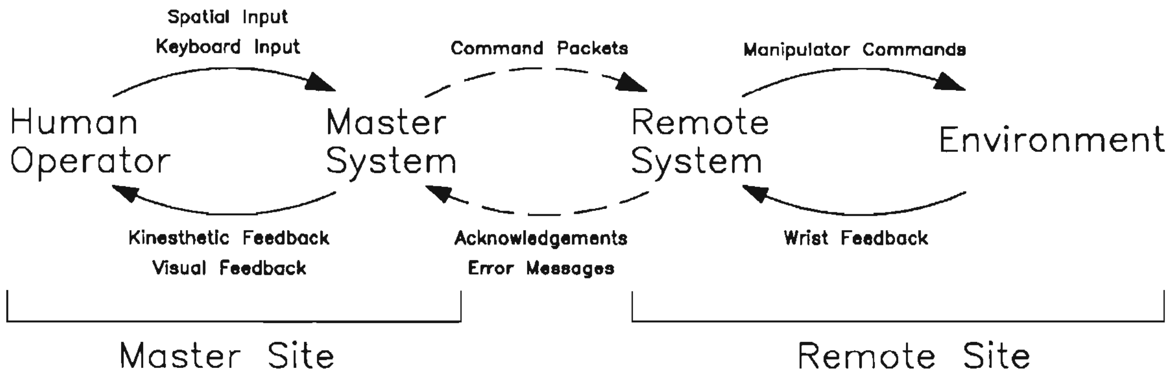


Figure 2.1: Basic Teleprogramming Schematic

loops are closed locally at both the master and remote sites. The implementation and interactions of the teleprogramming elements are shown in figures 2.1 and 2.2.

The human operator is the main decision making element in the teleprogramming system. The operator is responsible for task planning and error recovery. In the MERIONETTE system, the operator uses a Puma 250 manipulator as a spatial positioning device. The operator holds the end of the robot, which is backdriven to follow the operator's motions.

The Puma is also controlled to provide the operator with kinesthetic feedback from forces generated in the master model: a virtual remote site environment. Visual feedback is provided by a Silicon Graphics (SGI; Mountain View, CA) Iris workstation. The model is built from remote site data (video, laser scan, sonar scan, etc.) as the first step in a teleprogramming session. The interaction between the human operator and the graphical model is identical to classical bilateral teleoperation.

Input from the human operator and *a priori* information about the task allow the master system to automatically generate commands that are sent to the remote site. These commands are generated once every second unless a change in contact states is detected. In this case, a command is immediately generated.

The communication system links the master and remote systems together. Communication is currently limited to small command packets, called execution environments, sent from the master to the remote site at least once every second, and an acknowledgment or an error message sent back to the master at the completion of

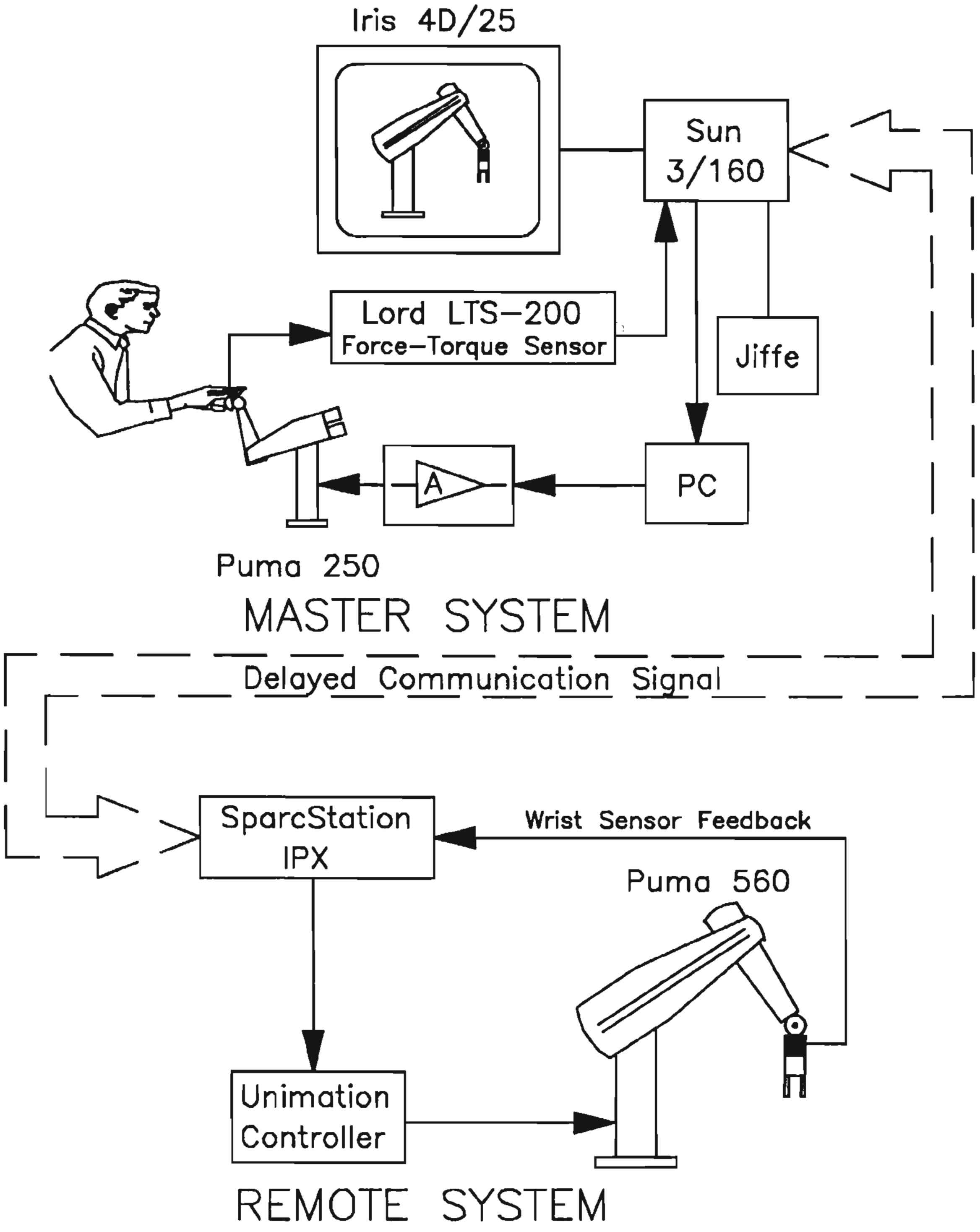


Figure 2.2: The experimental teleprogramming testbed

every command. There is a programmable communication time delay in the current implementation. Current experimentation uses a delay of 5 seconds.

When errors occur at the remote site, a message is sent back to the master system, and the graphics model is reset to correspond to the remote site error state. The operator must then decide how to continue with the task.

2.3 Limitations

Some basic limitations exist in the current implementation of teleprogramming. The most obvious is that the master model is not dynamic. It is impossible to run a meaningful dynamic simulation on an uncertain world model. Another limitation is that the motions of the master arm are quantized into straight line motions of a specific time length. In order for the remote system to work properly, this time length must be no shorter than the time needed by the remote site to parse and interpret an execution environment. Otherwise, the stream of commands sent to the remote site would continue to build. This limitation makes it difficult to follow curved trajectories.

2.4 Summary

A brief introduction to modern teleoperative systems has been presented, as well as an overview of the teleprogramming paradigm. Using teleprogramming, useful work can be accomplished in time-delayed environments. However, teleprogramming may also have uses in non-time-delayed environments. In the next chapter, the remote site robot system will be presented.

Chapter 3

The Remote System

Working underwater, in shallow space, or elsewhere, the remote system operates in a simple and methodical way. It receives a short set of commands that constitute an execution environment, and converts these commands into low level manipulator instructions which are then used to execute a single manipulator motion. These instructions are executed by the remote manipulator, and the execution is monitored for errors. The commands must be simple enough for the remote system to quickly and easily distinguish between successful and unsuccessful executions. The system is similar in this respect to supervisory control: the remote site robot system works as an autonomous agent with short-range goals.

The main tasks of the remote site system are:

- Parsing and interpretation of commands, including environment-specific parameters.
- Control of the remote site manipulator.
- Monitoring the remote process, including detection of errors.
- Reporting to the master system.

The remote robot system must work within a set of specifications in part imposed by the teleprogramming system. The remote system:

- must not unintentionally damage itself or the environment.
- must interpret and execute commands that may be inaccurate to within a specified tolerance.
- must interpret and execute commands from the master system that have only approximate (relative) magnitude information for certain parameters (eg. force, guards).
- must not build up an appreciable time lag between expected and actual command execution times.
- must be able to distinguish between successful and unsuccessful execution of commands.

The remote system developed for teleprogramming is insensitive to the communication time delay. It will work equally well with no time delay or a very long time delay. The remote system is in some sense independent of the teleprogramming paradigm: it can be directly controlled, controlled via a different teleoperation scheme, or follow a previously stored sequence of commands.

The remote system is implemented with a Puma 560 robot manipulator with a standard Unimation controller, and a SparcStation IPX computer. Position and force feedback is supplied by an instrumented compliant wrist. The remote manipulator uses two control loops. A low-level hybrid control loop is used to send the robot basic velocity commands at each controller interrupt (currently limited to 20ms), and to respond to sensor input. A higher level loop parses and interprets the command packets sent from the master site into low level velocity commands, and checks for errors in the execution of these commands.

Two phases of remote site implementation are presented here. The first phase, described briefly in this chapter, covers the basic implementation of the remote site system, including hardware, software, and the basic control structure. The successful demonstration of the teleprogramming system marked the end of Phase I. Phase II covers the refinement and improvement of the remote system, and is an ongoing effort.

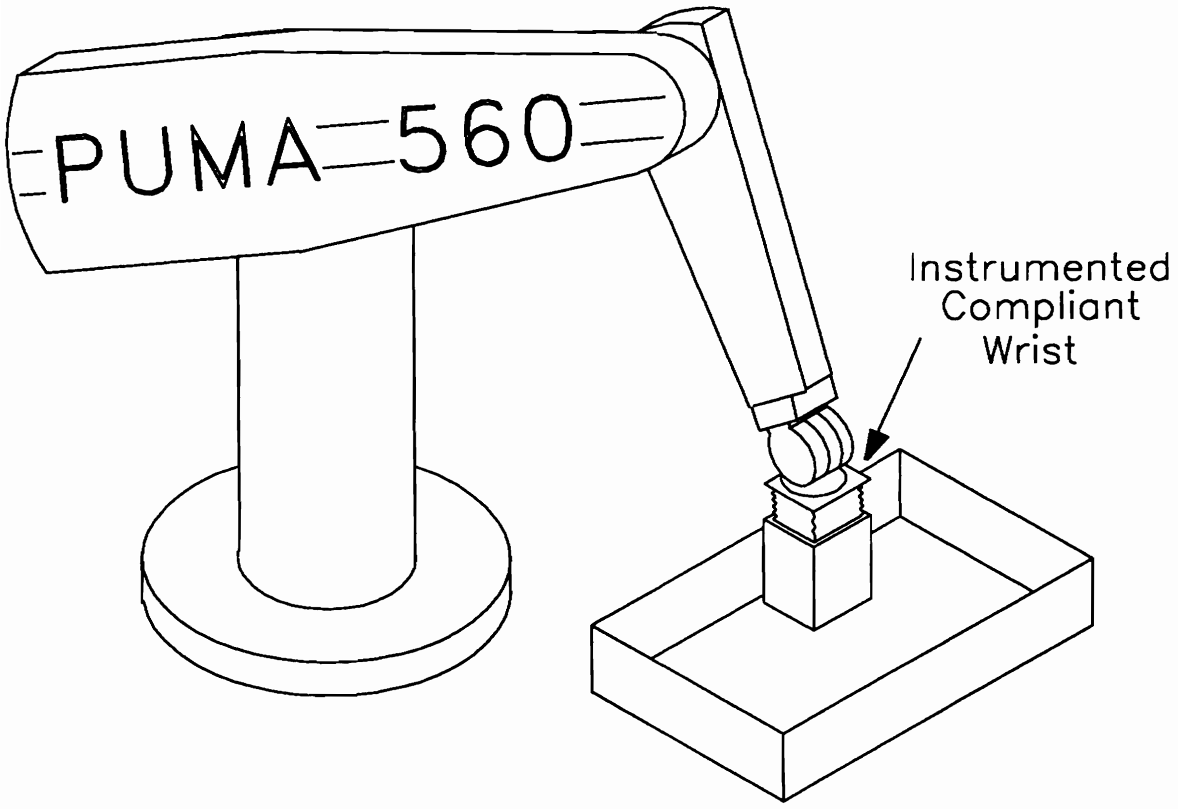


Figure 3.1: Preliminary Experimental Remote Site Environment

3.1 Preliminary Experimental Results

The effectiveness of the MERIONETTE system has demonstrated with a remote environment composed of an open box that can be explored by the system, as illustrated in figure 3.1. A typical experiment would be to move into contact with the bottom of the box, slide to a wall, and then slide into a corner. Motions and interactions with the environment were relatively simple, and the system was able to execute on the order of 100 commands successfully before errors occurred. Basic elements of the original (phase I) system were tested in this environment, and areas where further research was required were found. The remaining chapters of this paper document the start of phase II, describing areas of further research at the remote site system.

3.2 Remote Site Research Topics

Preliminary experimentation of the teleprogramming system indicated several topics for further research.

The low level control is based in an arbitrary task frame, although sensor feedback is based at the manipulator wrist. The usefulness of arbitrary frames became quite apparent early in the development of the system. However, the stability of hybrid control using arbitrary task frames has never been investigated. Implementation of the low-level hybrid control is presented in chapter 4, as well as a presentation of the use of arbitrary task frames for control. A simple study shows how sensor noise alone can cause instability when the task frame is located far enough from the wrist.

The higher-level control loop contains many research issues. The experiments show that, because of tolerance limits, small motions terminated by collisions with the environment (guarded motions) are not valid, and may result in large increases in the time lag of the remote system. Further, the experiments have shown that more research is necessary on motion-terminating conditions, as constant sensor limits did not work adequately. Depending on sensor-based and environment-based noise, the system could either mistake sensor noise for a collision, or vice-versa. A general overview of the parsing/interpreting loop will be presented in chapter 5, as well as details of the process of parsing guarded moves, and determination of motion-terminating conditions.

The need for manipulator tools to increase the manipulative capabilities of the system became apparent with the limitations of the preliminary system. Tool usage by the teleprogramming system will be explored in chapter 6.

3.3 Summary

A remote system for teleprogramming has been developed and refined with this research. Preliminary experimentation indicated limitations of the original remote system, and these limitations are examined and resolved with the research presented in following chapters.

This system was developed as the remote site for the teleprogramming concept. However, the compliance and flexible control characteristics of the system has proven quite useful for other applications [Campos 1992, Salganicoff 1992, Sinha 1991]. The semi-autonomous, compliant system incorporating hybrid position/force control may have many more potential applications, both in delayed and undelayed environments.

In the next three chapters, the remote site research topics introduced above will be examined in greater detail, starting with the servo-level hybrid control of the remote manipulator, continuing with the higher level command execution, and concluding with control and use of tools in the remote world.

Chapter 4

Hybrid Control Using an Instrumented Compliant Wrist

A robot manipulator that must work in an unknown, unstructured environment requires the ability to interact safely with the environment. In the highly structured industrial environment, a rigid manipulator using pure position control is adequate, and possibly desirable in order to achieve high speed motions. However, when wanted and unwanted collisions of the robot manipulator with the environment occur at inexact locations, a manipulator with some level of compliance, implemented with hardware, software, or both, is necessary. Furthermore, as the manipulator continues to move in various contact states with the environment, a hybrid controller with stable mode switching characteristics is needed.

The remote manipulator uses a low level proportional plus derivative feedback (PD) controller based on a hybrid position/force algorithm. Sensor feedback is provided with an instrumented compliant wrist, developed as part of this research. Implementation of the hybrid controller will be presented in this chapter.

The master system generates task frames specific to the current task conditions. Because the task frame in which the hybrid control is specified may be arbitrary, a discussion of the validity of completely arbitrary frames is presented here.

4.1 Background

Whitney [Whitney 1982] showed the usefulness of remote center of compliance (RCC) devices for peg-insertion tasks. The RCC device passively causes the peg to correctly compensate for forces resulting from misalignment in the insertion process. However, the RCC device is not ideal for generalized tasks [Lindsay et al. 1991, Yoshikawa 1990], and positional accuracy is lost with a passive compliant device [Volpe and Khosla 1991]. Xu developed an all-purpose 6 DOF instrumented compliant wrist to overcome these problems [Xu and Paul 1988]. Three main benefits arise from using an instrumented compliant wrist. The first is the ability to track surfaces, allowing the task frame to become dynamic with the end effector trajectory [Shutter and Brussel 1988]. Second, using the wrist as a force/torque sensor allows for more responsive force control [Roberts et al. 1985, Whitney 1985]. Finally, the transition from unconstrained to constrained motions are facilitated, and impact energy is absorbed [Volpe and Khosla 1991, Xu and Paul 1988].

Hybrid control was formalized by Raibert and Craig [Raibert and Craig 1981], and has its origins in compliance control [Mason 1981, Paul and Shimano 1976, Inoue 1971]. Natural and artificial constraints of a task dictate orthogonal axes that should be either position controlled or force controlled. Xu adapted the hybrid control scheme for controlling a manipulator with the instrumented compliant wrist as the force/torque sensor. Position feedback is recovered directly from the wrist. Force feedback is obtained implicitly from the stiffness matrix of the wrist.

Low level hybrid control using the instrumented compliant wrist is further generalized from the work of Xu by allowing control task frames to be arbitrarily defined. The effect of defining a task frame remote from the wrist sensor is examined.

Zhang has shown that hybrid control for a 6-DOF system is stable for a robot with three perpendicular axes intersecting at the wrist point, with the task frame located at this wrist point [Zhang 1986]. It will be shown herein that the system can become unstable because of sensor noise alone when the task frame is located remotely from the wrist point.

4.2 Hardware

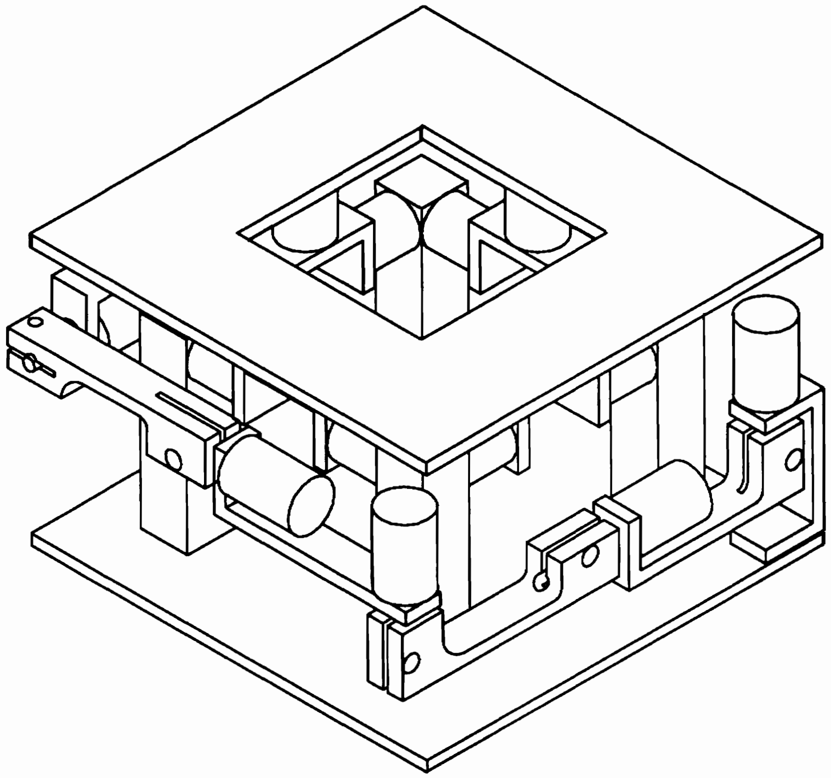


Figure 4.1: Instrumented Compliant Wrist

The remote manipulator is fitted with an instrumented compliant wrist (see figure 4.1). Compliance in a robot wrist is desirable in order to reduce the effect of impacts between the robot and the environment, and to provide for force control. Further, some compliance in the system will be beneficial to assembly and disassembly tasks with inaccurate positions. However, a compliant wrist limits the effective stiffness of the manipulator in position control, and the exact position of the end of the wrist (and thus the environment, when in contact) is lost [Volpe and Khosla 1991]. By instrumenting the wrist, both problems are overcome. Active control can increase the stiffness of the system, and the position transform of the wrist is known. Using the instrumented wrist as a compliant force/torque sensor leads to more responsive force control than with a stiff sensor [Roberts et al. 1985], and more accurate position control than with a compliant wrist [Xu 1989].

The two components of the wrist are the compliant structure, and the sensing linkage. The compliant structure is composed of rubber elements with known stiffness, connecting the upper and lower plates of the wrist. The sensing linkage is composed of six links, with potentiometers at their connections. By reading the change in voltage across the potentiometers, the angle of each joint can be determined. Using a simple forward kinematic formulation, the relative position of the top plate with respect to the bottom plate can be found. This displacement is used directly for positional feedback, and multiplied by the compliance matrix of the wrist, it provides implicit force feedback. More information about the instrumented compliant wrist can be found in [Lindsay and Paul 1991].

4.3 Hybrid Control

The remote manipulator uses a hybrid force/position controller. Displacements from the wrist sensor are used directly for position feedback. Force feedback is determined from the wrist stiffness matrix. A conceptual organization of the controller used is shown in figure 4.2. The main difference between this control system and Raibert and Craig [Raibert and Craig 1981] is that the force and position feedback both use a common sensory system, the instrumented wrist. Both force and positional feedback are ultimately used to change the manipulator trajectory with position offsets. The sensor signal is transformed from the wrist frame TW (the sensor coordinate frame) into the task frame TF[1] for control:

$$ {}^{TF}\mathbf{X} = \left[{}^{TF}_{TW}\Lambda\right] \left[{}^{TW}\mathbf{X}\right] \tag{4.1} $$

$$ {}^{TF}\mathbf{F} = \left[{}^{TF}_{TW}T\right] K \left[{}^{TW}\mathbf{X}\right] \tag{4.2} $$

where:

[1]Raibert and Craig [Raibert and Craig 1981] use the notation C for the task frame (control frame). The notation here differs from [Raibert and Craig 1981] in many places. Note especially that the parameters for s_j (mode selection) are reversed from [Raibert and Craig 1981]

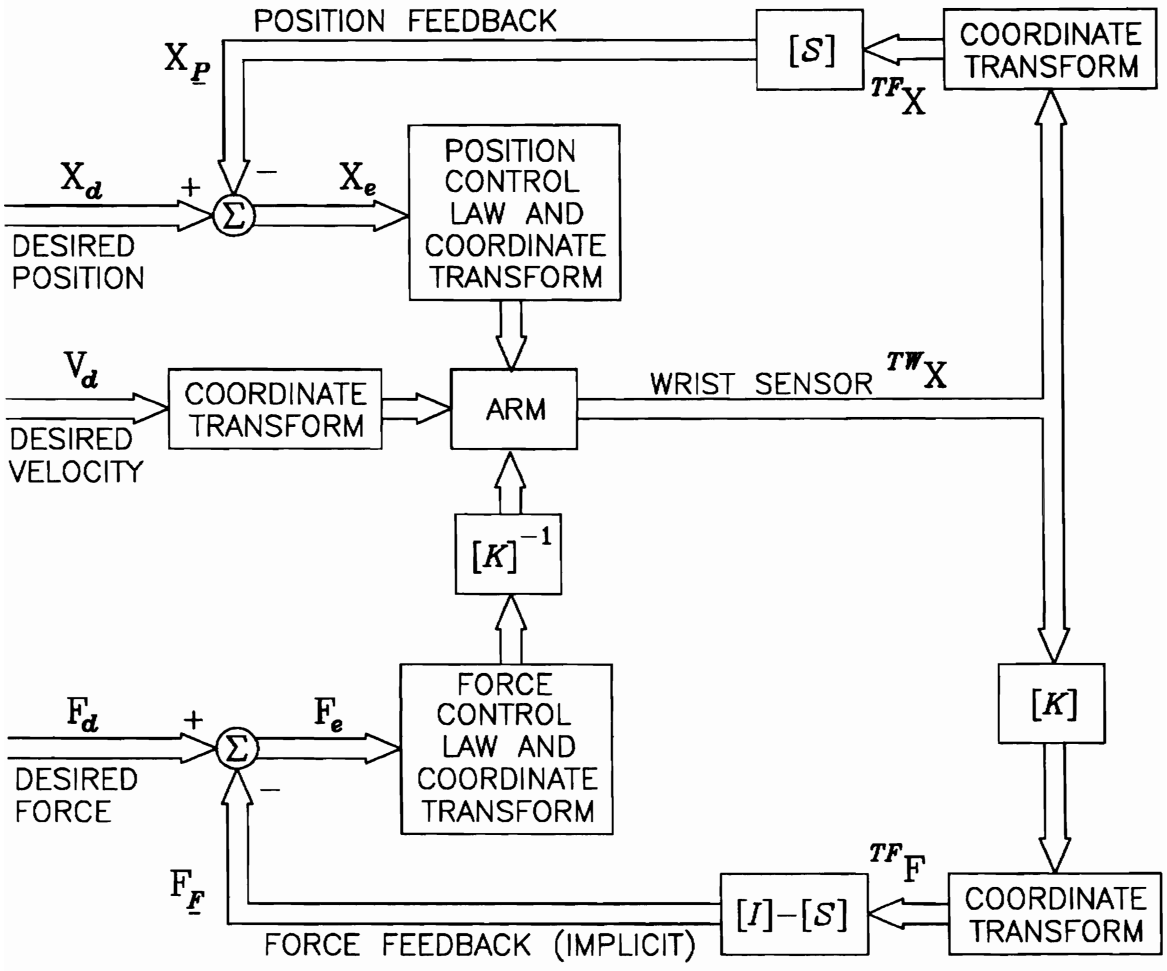


Figure 4.2: Conceptual Organization of Hybrid Controller

${}^{TW}\mathbf{X}$ = wrist displacement, $[\delta x, \delta y, \delta z, \delta\theta_x, \delta\theta_y, \delta\theta_z]^T$

$$\left[{}^{TF}_{TW}\Lambda\right] = \begin{bmatrix} \left[{}^{TF}_{TW}R\right] & \vdots & [\mathbf{P}\times]\left[{}^{TF}_{TW}R\right] \\ \cdots & \cdot & \cdots \\ \mathbf{0}_{3\times3} & \vdots & \left[{}^{TF}_{TW}R\right] \end{bmatrix}$$

= kinematic transformation from TW to TF

$$\left[{}^{TF}_{TW}T\right] = \begin{bmatrix} \left[{}^{TF}_{TW}R\right] & \vdots & \mathbf{0}_{3\times3} \\ \cdots & \cdot & \cdots \\ [\mathbf{V}\times]\left[{}^{TF}_{TW}R\right] & \vdots & \left[{}^{TF}_{TW}R\right] \end{bmatrix}$$

= force transformation from TW to TF

$\left[{}^{TF}_{TW}R\right] = 3 \times 3$ rotation matrix from TW to TF

$$[\mathbf{P}\times] = \begin{bmatrix} 0 & -P_z & P_y \\ P_z & 0 & -P_x \\ -P_y & P_x & 0 \end{bmatrix}$$

$\mathbf{P}$ = vector from the origin of TW to the origin of TF expressed in TW

$$[\mathbf{V}\times] = \begin{bmatrix} 0 & -V_z & V_y \\ V_z & 0 & -V_x \\ -V_y & V_x & 0 \end{bmatrix}$$

$\mathbf{V}$ = vector from the origin of TF to the origin of TW expressed in TF

$K = 6 \times 6$ Stiffness matrix for the wrist sensor

${}^{TF}\mathbf{X}$ = position feedback in task frame coordinates

${}^{TF}\mathbf{F}$ = force feedback in task frame coordinates

At this point, the feedback is partitioned into position feedback and (implicit) force feedback. The mode selection vector $\mathbf{S}$ specifies which degrees of freedom are to be force controlled (indicated by $s_j = 0$) and which are to be position controlled ($s_j = 1$). The mode selection matrix is defined by:

$$[\mathcal{S}] = \text{diag}(\mathbf{S}) = \begin{bmatrix} S_1 & & & & 0 \\ & S_2 & & & \\ & & S_3 & & \\ & & & \cdots & \\ 0 & & & & S_6 \end{bmatrix}$$

Partitioned force and position feedback are found by:

$$\mathbf{X}_{\underline{P}} = [\mathcal{S}] \left[{}^{TF}\mathbf{X} \right] \tag{4.3}$$

$$\mathbf{F}_{\underline{F}} = ([I] - [\mathcal{S}]) \left[{}^{TF}\mathbf{F} \right] \tag{4.4}$$

Finally, the error signals are found as:

$$\mathbf{X}_e = \mathbf{X}_d - \mathbf{X}_{\underline{P}} \tag{4.5}$$

$$\mathbf{F}_e = \mathbf{F}_d - \mathbf{F}_{\underline{F}} \tag{4.6}$$

where the subscript d denotes the desired values.

Position and force errors are multiplied by proportional and derivative gains:

$$\begin{aligned} \mathbf{X}_c &= [K_{pp}]\mathbf{X}_e + [K_{pd}]\dot{\mathbf{X}}_e && (4.7) \\ \mathbf{F}_c &= [K_{fp}]\mathbf{F}_e + [K_{fd}]\dot{\mathbf{F}}_e && (4.8) \\ \mathbf{V}_c &= \mathbf{V}_d && (4.9) \end{aligned}$$

where:

$[K_{pp}], [K_{pd}], [K_{fp}]$, and $[K_{fd}] = 6 \times 6$ proportional and derivative feedback matrices for position and force directions

$\mathbf{V}_d$ = desired velocity

$\dot{\mathbf{X}}_e$ and $\dot{\mathbf{F}}_e$ = derivative feedback found from $\mathbf{X}_{e,i} - \mathbf{X}_{e,i-1}$ and $\mathbf{X}_{e,i} - \mathbf{X}_{e,i-1}$

These control signals are transformed back into the wrist frame:

$$^{TW}\mathbf{X}_s = \left[^{TW}_{TF}\Lambda\right] \mathbf{X}_c \tag{4.10}$$

$$^{TW}\mathbf{F}_s = [K]^{-1} \left[^{TW}_{TF}T\right] \mathbf{F}_c \tag{4.11}$$

$$^{TW}\mathbf{V}_s = \left[^{TW}_{TF}\Lambda\right] \mathbf{V}_c \tag{4.12}$$

where:

$[K]^{-1}$ = inverse of wrist stiffness matrix

These signals are next cast into matrix form $^{TW}\mathcal{X}_s$, $^{TW}\mathcal{F}_s$, and $^{TW}\mathcal{V}_s$, and converted into joint space:

$$\mathbf{q}_p = [J]^{-1} \left[^{TW}\mathcal{X}_s\right] \tag{4.13}$$

$$\mathbf{q}_f = [J]^{-1} \left[^{TW}\mathcal{F}_s\right] \tag{4.14}$$

$$\mathbf{q}_v = [J]^{-1} \left[^{TW}\mathcal{V}_s\right] \tag{4.15}$$

where:

$[J]^{-1}$ = Inverse Jacobian of the manipulator

Robot joint angles are moved with velocity input and also comply with force errors:

$$(\theta_{des})_i = (\theta_{des})_{i-1} + (\mathbf{q}_v)_i + (\mathbf{q}_f)_i \tag{4.16}$$

The robot must also compensate for deflection in the wrist for positional control:

$$(\theta_{req})_i = (\theta_{des})_i + (\mathbf{q}_p)_i \tag{4.17}$$

The joint angles $(\theta_{req})_i$ are updated and sent to the robot joint controller each interrupt (20ms). The joint controller has its own position feedback control which servos at a much higher rate.

4.4 Remote Frames

Any cartesian frame can be partitioned into force and position directions for the hybrid control. Two obvious choices for the frame are the tool coordinate frame ($T6$), and a world coordinate frame with origin at the end of the manipulator. However, control is not limited to these two frames. Any frame relative to the $T6$ frame or the world coordinate frame can be used.

Figure 4.3 shows some examples of using remote task frames. In 4.3 (a), motion for turning a crank can be specified in a frame fixed to the crank with origin at the manipulator wrist (frame C), or a task frame aligned with the axis of rotation of the wrist (frame TF). In terms of control, turning the crank in each frame can be specified as:

Frame C		Frame TF	
Position	Force	Position	Force
	$f_x = 0$		$f_x = 0$
$v_y = p_1$			$f_y = 0$
	$f_z = 0$		$f_z = 0$
	$\tau_x = 0$		$\tau_x = 0$
	$\tau_y = 0$		$\tau_y = 0$
$\omega_z = 0$		$\omega_z = \alpha_1$	

In both cases, the specified motion is valid throughout the motion of the crank. Specifying the motion in frame C, motion is given in the tangential direction to the rotation of the crank. Compliant motion control is needed to keep the manipulator on a circular trajectory and to compensate for any discrepancies between the environment and the model used to generate the trajectory. By specifying the motion in the task frame TF, the motion of turning the crank becomes a simple rotation about the natural axis of rotation of the crank. The compliance of the system is only needed to compensate for model errors.

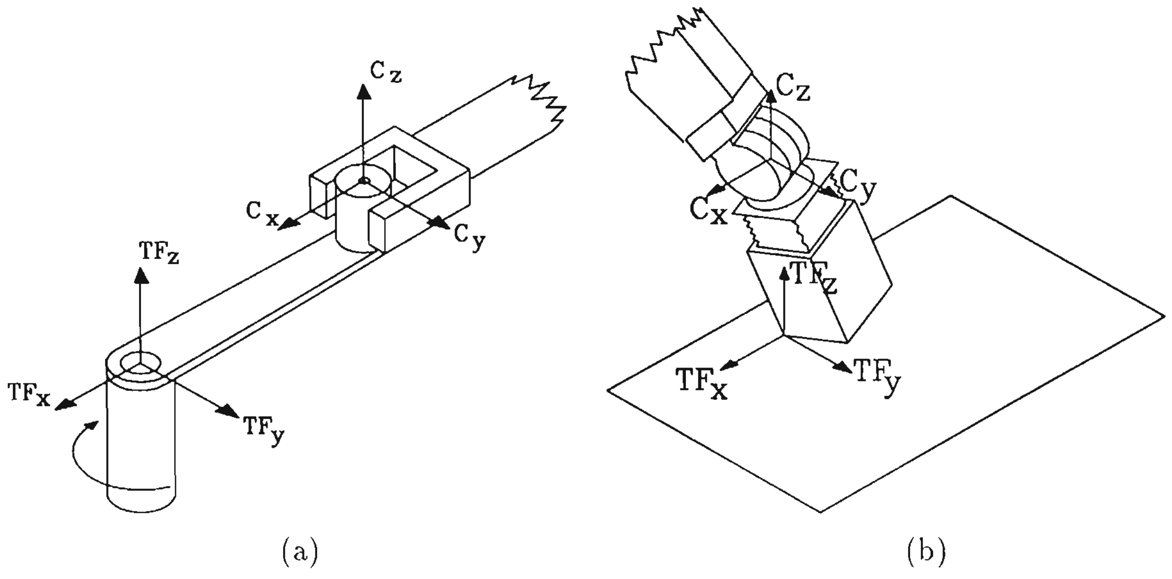


Figure 4.3: Remote Task Frames

In 4.3 (b), motion for pivoting a box into face-face contact with a surface is specified in a frame with origin at the wrist point of the robot, with the z-axis aligned with the surface normal (frame C), or a remote task frame with origin at the point of contact, and the z-axis aligned with the surface normal (frame TF). Motion is specified as:

Frame C		Frame TF	
Position	Force	Position	Force
$v_x = 0$		$v_x = 0$	
$v_y = 0$		$v_y = 0$	
	$f_z = -f_1$		$f_z = -f_1$
$\omega_x = -\alpha_1$		$\omega_x = -\alpha_1$	
$\omega_y = 0$		$\omega_y = 0$	
$\omega_z = 0$		$\omega_z = 0$	

In this example, both cases require the same compliant motion specification. However, unless surface friction is high, the motion in task frame C will cause the contact

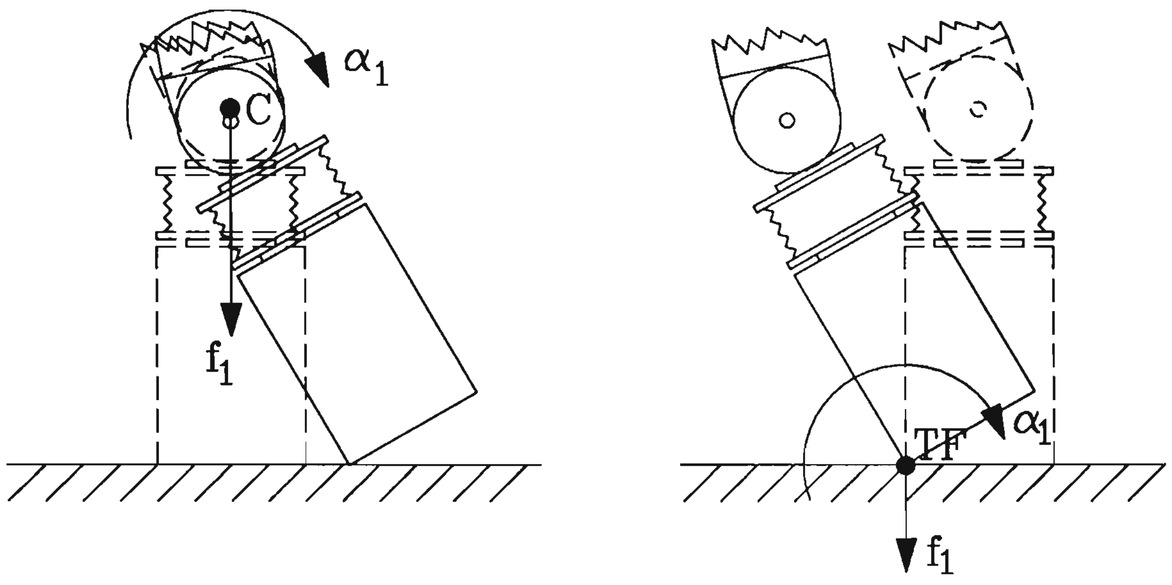


Figure 4.4: Rotations about frames C and TF

point to change, while the compliant control compensates for the changing normal forces. Motion in task frame TF will maintain the position of the contact point (see figure 4.4).

From these examples, it is evident that at certain times the use of a remote task frame is desirable. However, there are drawbacks to using frames that are located far from the wrist point of the robot.

The wrist sensor is located at the end of the robot. Control about a frame with an origin that is far from the end of the robot can become unstable due to the amplification of sensor noise. Also, small motions in the task frame are transformed into large motions at the end of the manipulator. The combination of sensor noise amplification and large motions of the robot presents a natural limit to the distance that the task frame can be located from the end of the manipulator. However, decreasing the system gains can make this motion stable

A brief study of the planar 3 degree of freedom manipulator shown in figure 4.5 illustrates one of the more basic problems with using a remote frame for control. In this simple manipulator, the sensors are located at the origin of the frame TW. The control loop for controlling this manipulator in the frame located at the end of the

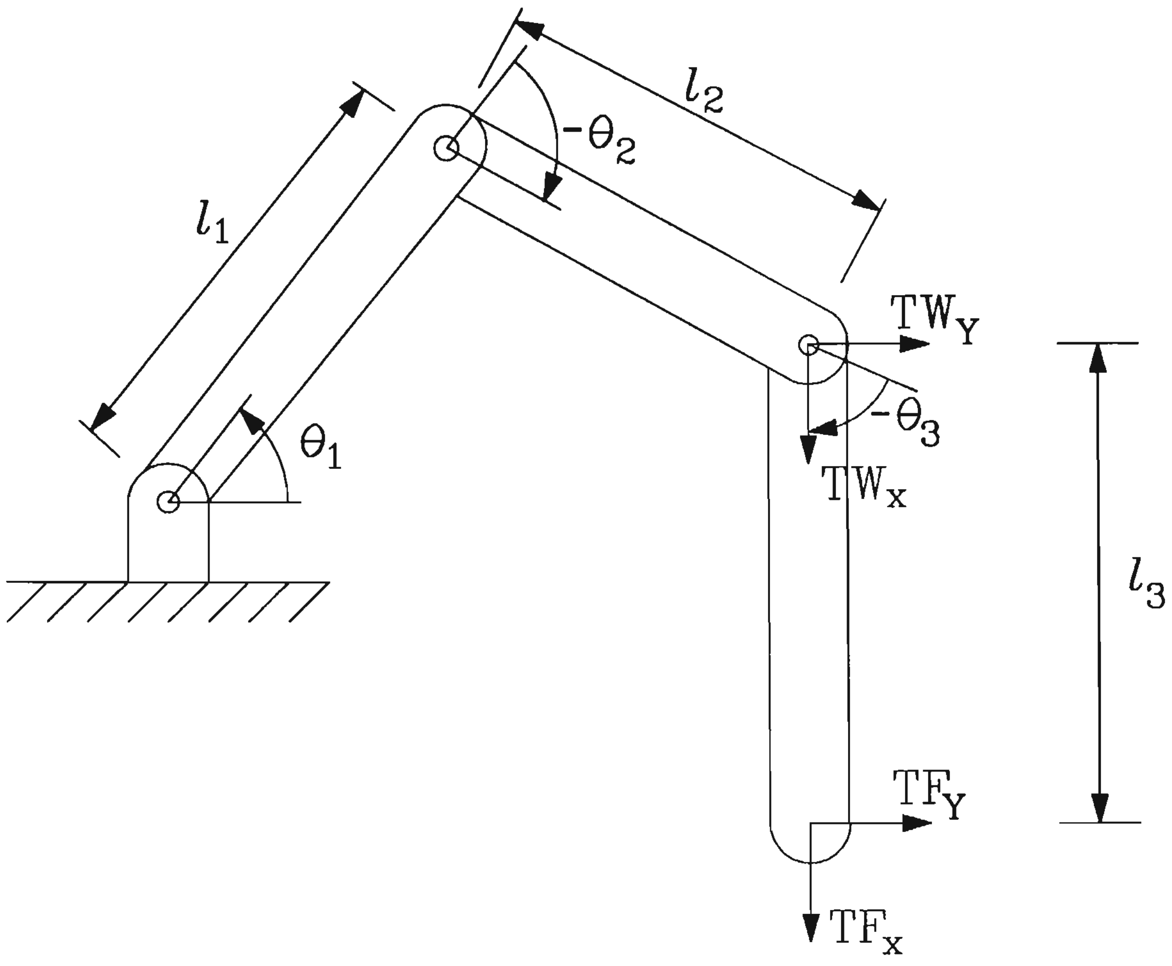


Figure 4.5: Three degree of freedom robot with long final link

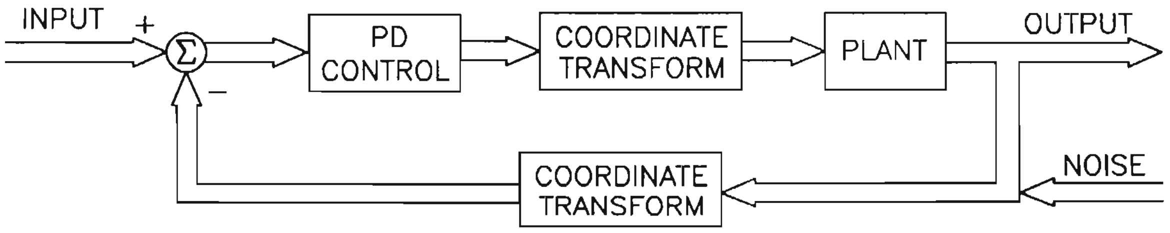


Figure 4.6: Simplified Control

final link (TF) is simplified from the general hybrid control presented previously, and is shown in figure 4.6. To transform the sensor readings from the sensor frame (TW), a linearized coordinate transform is used as shown below:

$$\begin{bmatrix} \delta x_{TF} \\ \delta y_{TF} \\ \delta\theta_{TF} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & l_3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_{TW} \\ \delta y_{TW} \\ \delta\theta_{TW} \end{bmatrix} \tag{4.18}$$

To transform back to the wrist frame,

$$\begin{bmatrix} \delta x_{TW} \\ \delta y_{TW} \\ \delta\theta_{TW} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -l_3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_{TF} \\ \delta y_{TF} \\ \delta\theta_{TF} \end{bmatrix} \tag{4.19}$$

If there is no input except for sensor noise (see figure 4.7), the control loop can be compressed so that:

$$\mathbf{X}_d = \left[{}^{TW}_{TF}T\right] \mathcal{K} \left[{}^{TF}_{TW}T\right] \left[{}^{TF}\mathbf{X}\right] \tag{4.20}$$

where:

$\mathbf{X}_d$ = desired position (input to PLANT block) = $[\delta x_d, \delta y_d, \delta\theta_d]^T$

${}^{TF}\mathbf{X}$ = noise - output

$$\mathcal{K}_{3\times 3} = \text{gain matrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_\theta \end{bmatrix}$$

Substituting equation 4.18 for $\left[{}^{TF}_{TW}T\right]$ and equation 4.19 for $\left[{}^{TW}_{TF}T\right]$ leads to:

$$\begin{bmatrix} \delta x_d \\ \delta y_d \\ \delta\theta_d \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & (k_y - k_\theta)l_3 \\ 0 & 0 & k_\theta \end{bmatrix} \begin{bmatrix} \delta x_{TF} \\ \delta y_{TF} \\ \delta\theta_{TF} \end{bmatrix} \tag{4.21}$$

It can be seen that increasing the distance l_3 increases the off diagonal term, which can destabilize the system. A simple solution for this particular problem is the equate k_y and k_θ. However, this solution does not adapt to the 6-DOF system.

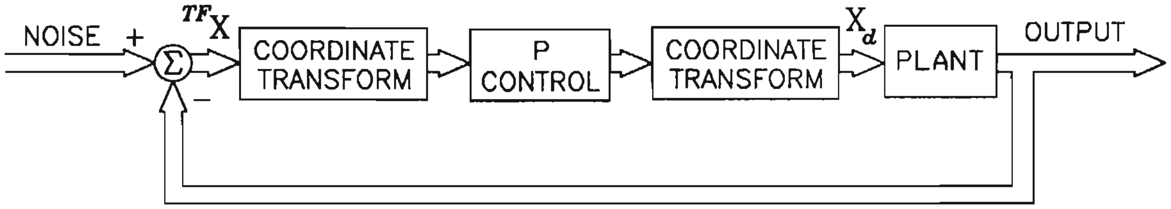


Figure 4.7: Noise Input

Although the formulation of this problem for the 6-DOF manipulator is much more complex, increasing the distance of the remote task frame has the same effect of effectively increasing the control gains, which destabilize the system. A simulation of the 3-DOF system is presented below, and experimentation on the remote system follows.

4.5 Simulation of 3-DOF Arm

A simulation of the 3-DOF arm has been constructed, using SIMULAB (The Math Works, Inc., Natick, Mass). The purely kinematic simulation has no input except for white noise. Note that the effect of dynamics is not included in this simulation. However, dynamic effects would further destabilize the system, as dynamic motions would effectively increase the input "noise" signal, unless the system were effectively damped.

Model parameters are shown in table 4.1.

Figures 4.8 and 4.9 show the model output with stable gains and $l_3 = 0$.

Figure 4.10 shows the simulation with the final link length still zero, but an off-diagonal gain introduced into the position of $(k_y - k_\theta)l_3$. When this term is increased to 120, the system becomes unstable. For this plot, the value of the off-diagonal term is 110.

The displacement of the task frame from the wrist sensor can be shown to be similar to increasing the off-diagonal term. Figure 4.11 shows the simulation with l_3 increased to 120. Here, the system is again nearly unstable. However, from the above analysis, the system should remain stable for l_3 as long as 220. The lack of stability

l_1	4.0
l_2	4.0
$\theta_{1,i}$	30°
$\theta_{2,i}$	-60°
$\theta_{3,i}$	-60°
k_x	1.0
k_y	1.0
k_θ	0.5
noise (trans.) (variance)	0.1
noise (rotat.) (variance)	0.01

Table 4.1: 3-DOF Model Parameters

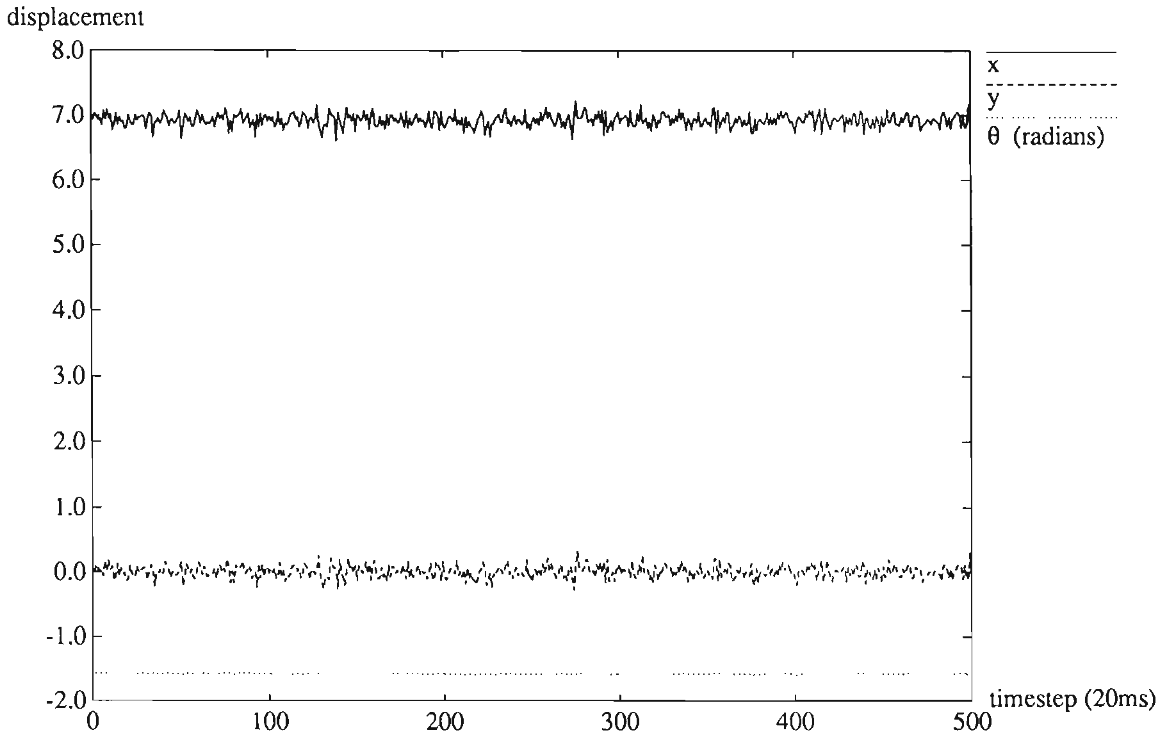


Figure 4.8: Cartesian Position, $l_3 = 0$

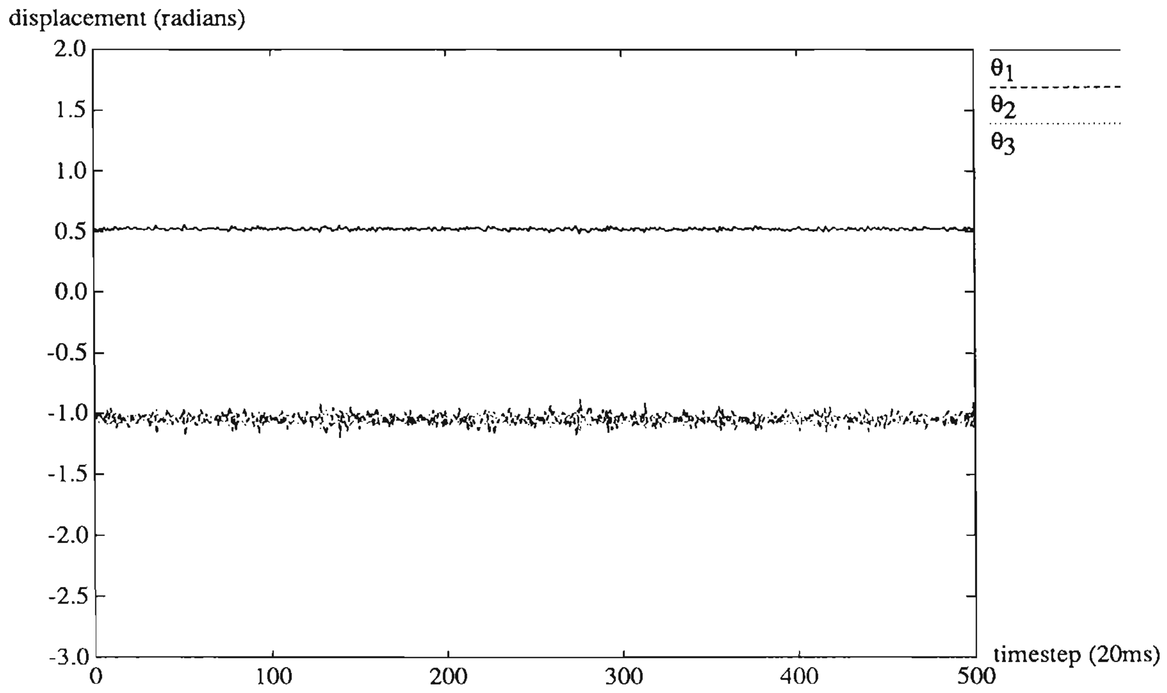


Figure 4.9: Joint Angles, $l_3 = 0$

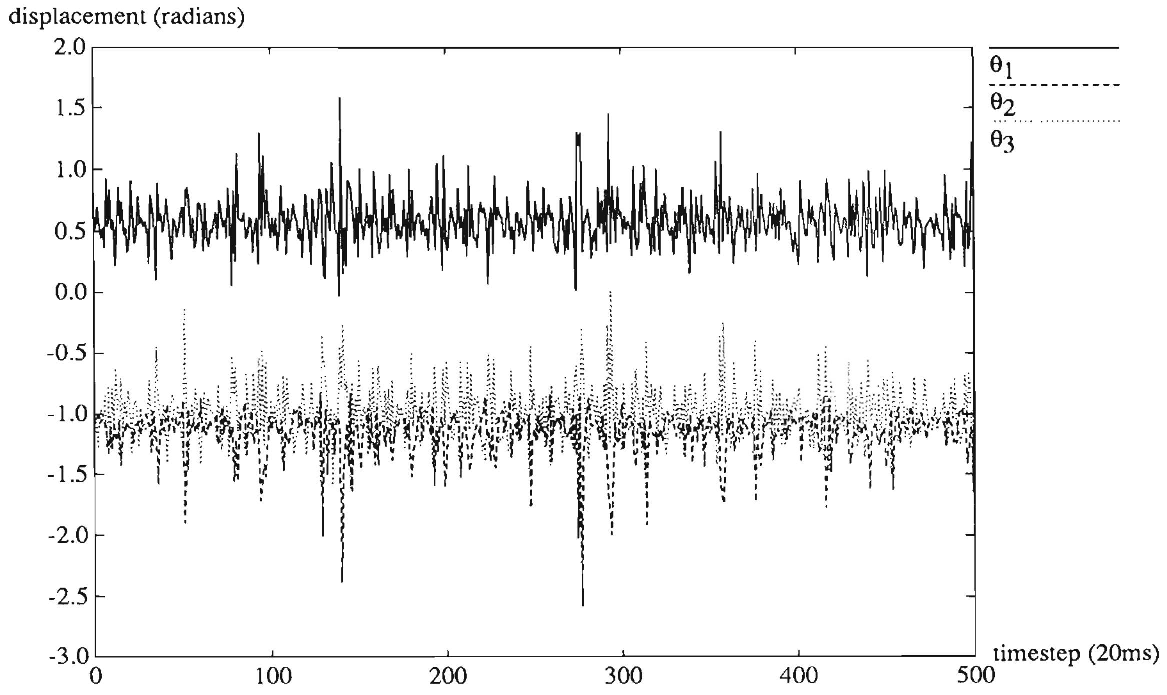


Figure 4.10: Joint Angles, Off-Diagonal Term = 110

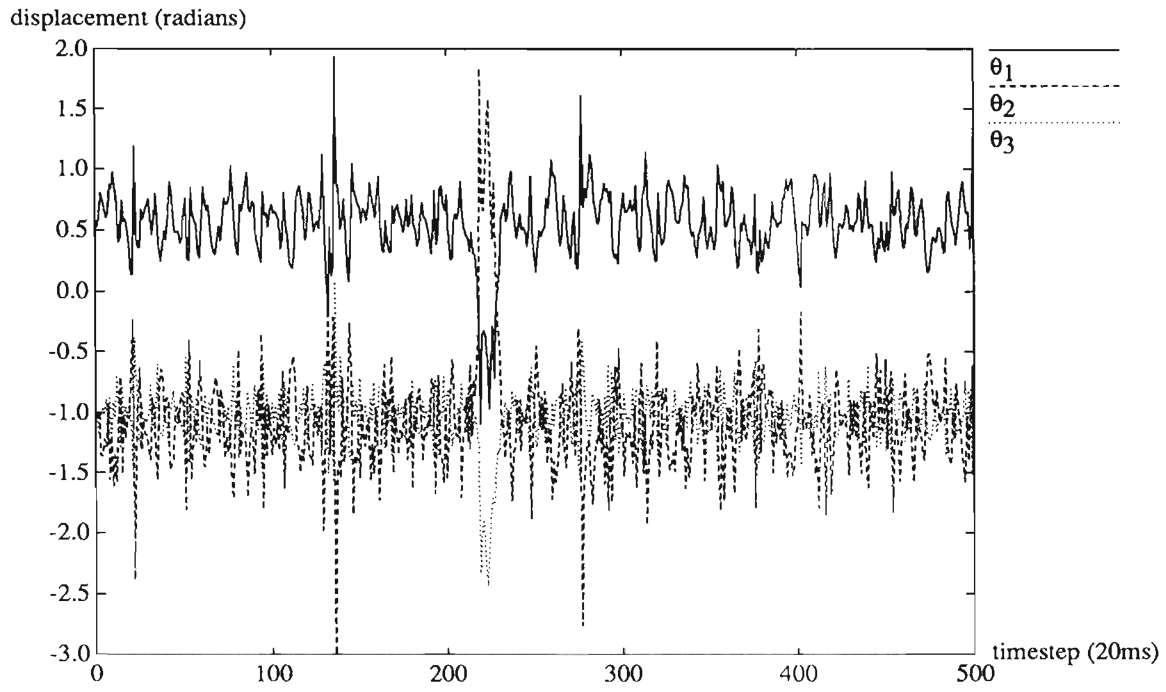


Figure 4.11: Joint Angles, l_3 = 120

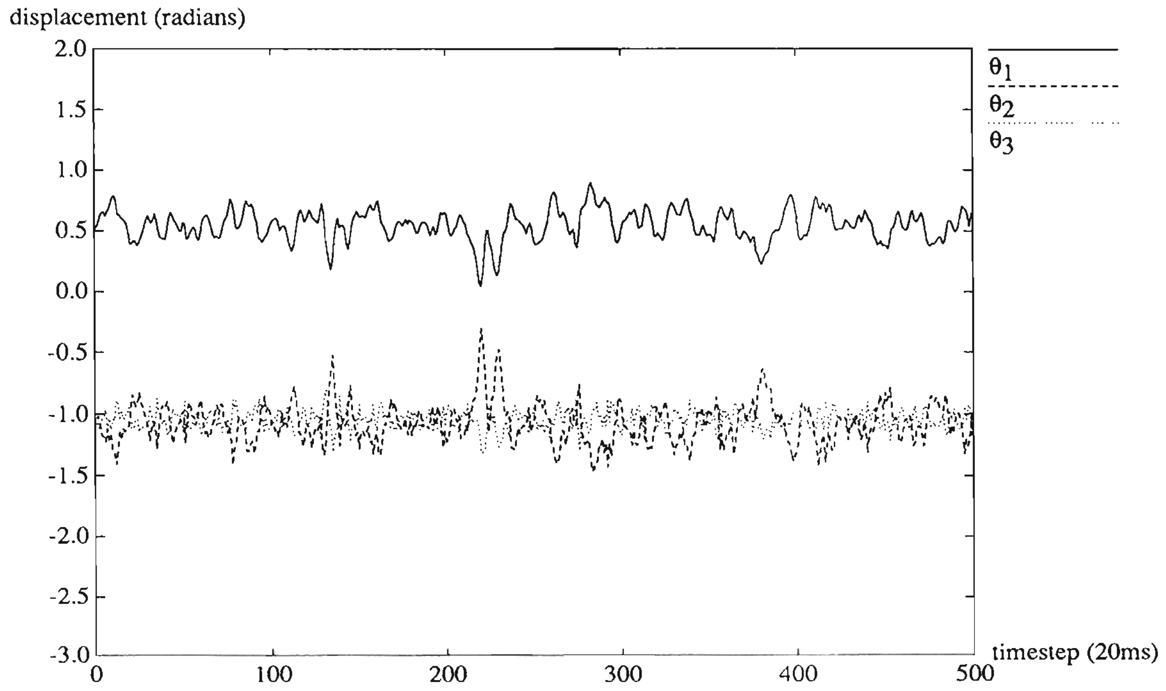


Figure 4.12: Joint Angles, l_3 = 120, Reduced Gains

is attributed to the non-linear terms that exist in the simulation.

The system stability can be restored by reducing the gain values. However, this will degrade the system response time. Figure 4.12 shows the system simulation with $l_3 = 120$, but the gains reduced by half. This system is considerably more stable.

Local stability of the 3-link planar manipulator is studied empirically by computing the closed-loop eigenvalues of the linearized system about the starting position in the above simulation. The procedure is similar to that followed in [An and Hollerbach 1987]. Only positional control is addressed here ($S = I$ in [An and Hollerbach 1987]), so the results are compatible with the results of [Fisher and Mujtaba 1992]. Assuming negligible joint velocities, the closed-loop system is described as

$$\delta\dot{x} = A\delta x \tag{4.22}$$

where $\delta x = (\delta q, \delta\dot{q})$, the joint angles and joint velocities, and the inertia matrix A is described below. To convert to cartesian coordinates,

$$\begin{aligned} \delta q &= J^{-1}K\delta x' & (4.23)\\ \delta\dot{q} &= \frac{d\left[J^{-1}\right]}{dt}K\delta x' + J^{-1}K\delta\dot{x}' & (4.24) \end{aligned}$$

Noting that

$$\frac{d\left[J^{-1}\right]}{dt} = -J^{-1}\dot{J}J^{-1} \tag{4.25}$$

However, the matrix $\dot{J} \rightarrow \mathbf{0}$ for negligible joint velocities. Substitution into 4.22 yields

$$\begin{aligned} \delta\dot{x'} &= \left(K^{-1}JAJ^{-1}K + K^{-1}\dot{J}J^{-1}K\right)\delta x' \\ &= K^{-1}JAJ^{-1}K\delta x' \\ &= A'\delta x' & (4.26) \end{aligned}$$

where $\delta x' = (\delta x, \delta\dot{x})$, the cartesian position and velocity, and

$$K = \begin{bmatrix} k_{px} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{py} & (k_{py} - k_{p\theta})\, l_3 & 0 & 0 & 0 \\ 0 & 0 & k_{p\theta} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{vx} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{vy} & (k_{vy} - k_{v\theta})\, l_3 \\ 0 & 0 & 0 & 0 & 0 & k_{v\theta} \end{bmatrix} \tag{4.27}$$

$$J = \left[\begin{matrix} -(l_1S_1 + l_2S_{12} + l_3S_{123}) & -(l_2S_{12} + l_3S_{123}) & -l_3S_{123} \\ (l_1C_1 + l_2C_{12} + l_3C_{123}) & (l_2C_{12} + l_3C_{123}) & l_3C_{123} \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}\right.$$
$$\left.\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -(l_1S_1 + l_2S_{12} + l_3S_{123}) & -(l_2S_{12} + l_3S_{123}) & -l_3S_{123} \\ (l_1C_1 + l_2C_{12} + l_3C_{123}) & (l_2C_{12} + l_3C_{123}) & l_3C_{123} \\ 1 & 1 & 1 \end{matrix}\right] \tag{4.28}$$

$$A = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -M^{-1} & -M^{-1} \end{bmatrix} \tag{4.29}$$

Assuming that the links have a homogeneous mass distribution, the inertia matrix for the 3-link manipulator is defined by:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{4.30}$$

$$m_{11} = I_1 + I_2 + I_3 + \frac{1}{4}\left(m_1l_1^2 + m_2l_2^2 + m_3l_3^2\right) + m_2\left(l_1^2 + l_1l_2C_2\right) +$$

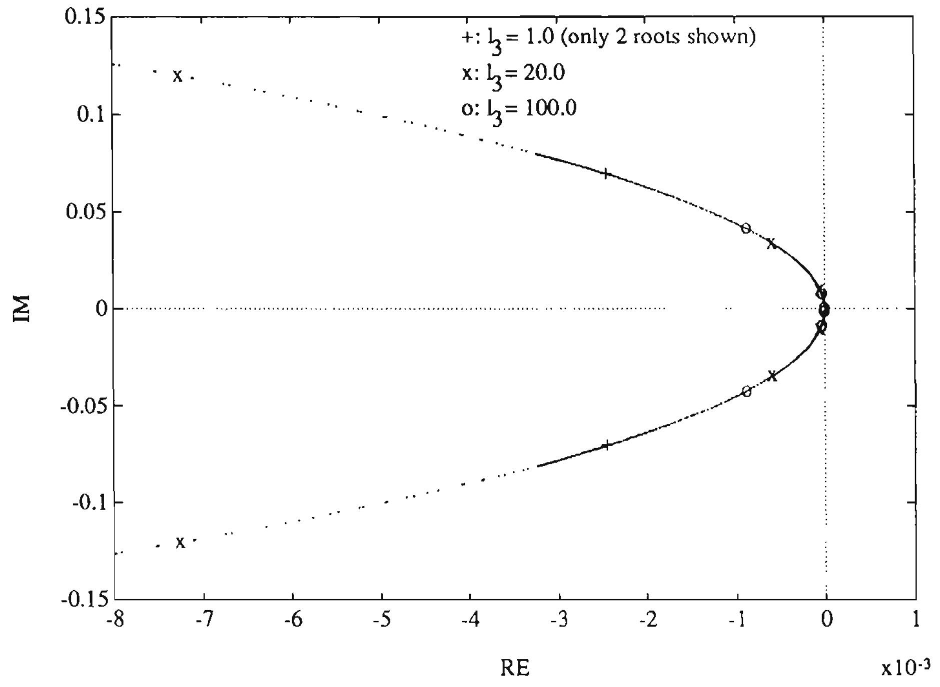


Figure 4.13: Root locus for 3-link manipulator

$$\begin{aligned}
& m_3\left(l_1^2+l_2^2+2l_1l_2C_2+l_1l_3C_{23}+l_2l_3C_3\right) && (4.31)\\
m_{22} &= I_2+I_3+\frac{1}{4}\left(m_2l_2^2+m_3l_3^2\right)+m_3\left(l_2^2+l_2l_3C_3\right) && (4.32)\\
m_{33} &= I_3+\frac{1}{4}m_3l_3^2 && (4.33)\\
m_{12} &= m_{21}=I_2+I_3+\frac{1}{4}\left(m_2l_2^2+m_3l_3^2\right)+\frac{1}{2}m_2l_1l_2C_2+ \\
& m_3\left(l_2^2+l_1l_2C_2+\frac{1}{2}l_1l_3C_{23}+l_2l_3C_3\right) && (4.34)\\
m_{13} &= m_{31}=I_3+\frac{1}{4}m_3l_3^2+\frac{1}{2}m_3l_3\left(l_1C_{23}+l_2C_3\right) && (4.35)\\
m_{23} &= m32=I_3+\frac{1}{4}m_3l_3^2+\frac{1}{2}m_3l_2l_3C_3 && (4.36)
\end{aligned}$$

Using the data from the simulation above, the eigenvalues A' can be found for varying lengths of l_3. The root locus is shown in figures 4.13 and 4.14. For large values of l_3, the eigenvalues are located very close to the origin, and the system is marginally stable. Additional unmodelled elements of the system will cause the roots to move into the right half plane, causing instability.

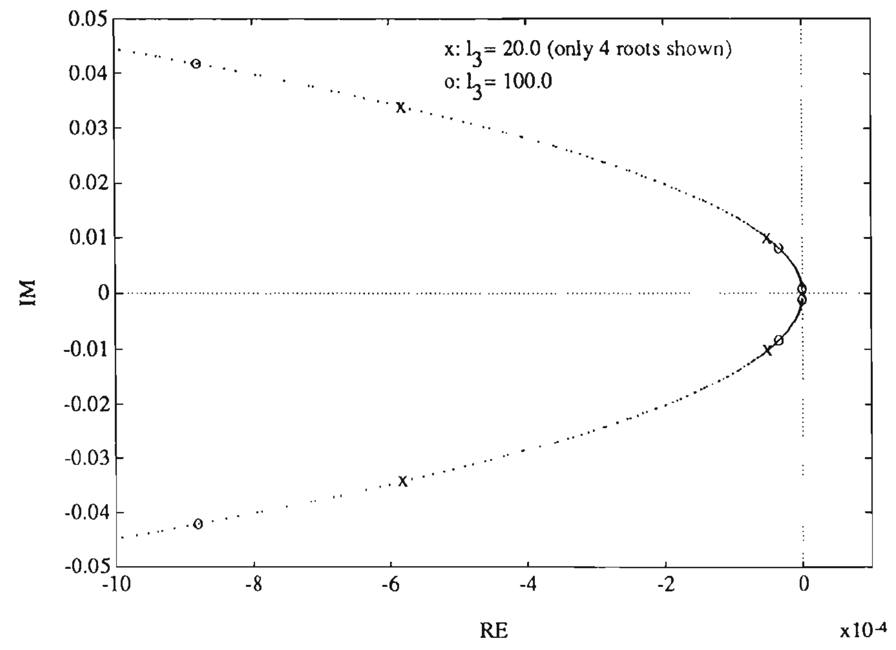


Figure 4.14: Close-up of root locus for 3-link manipulator

4.6 Experimental Results

Using the teleprogramming remote site system, experiments were conducted using remote task frames. The hybrid control works well using task frames located near the wrist sensor of the robot. As the task frame becomes more and more remote, however, the performance degrades.

Instability in the 3-DOF model was defined as when joint angles deviate sharply from the desired values, and do not return to the desired values. For safety reasons, instability in the physical robot system is interpreted only as significant motions away from the desired position. With the task frame located at the robot wrist, the system becomes unstable when the corresponding off-diagonal term of the gain matrix increases above 30,000. This gain is large because the noise level of the wrist sensor is much smaller than in the 3-DOF model above, and the addition of derivative gains in the control loop adds stability to the system. Figure 4.15 shows the cartesian deflections for this experiment. Notice that all three cartesian directions are effected

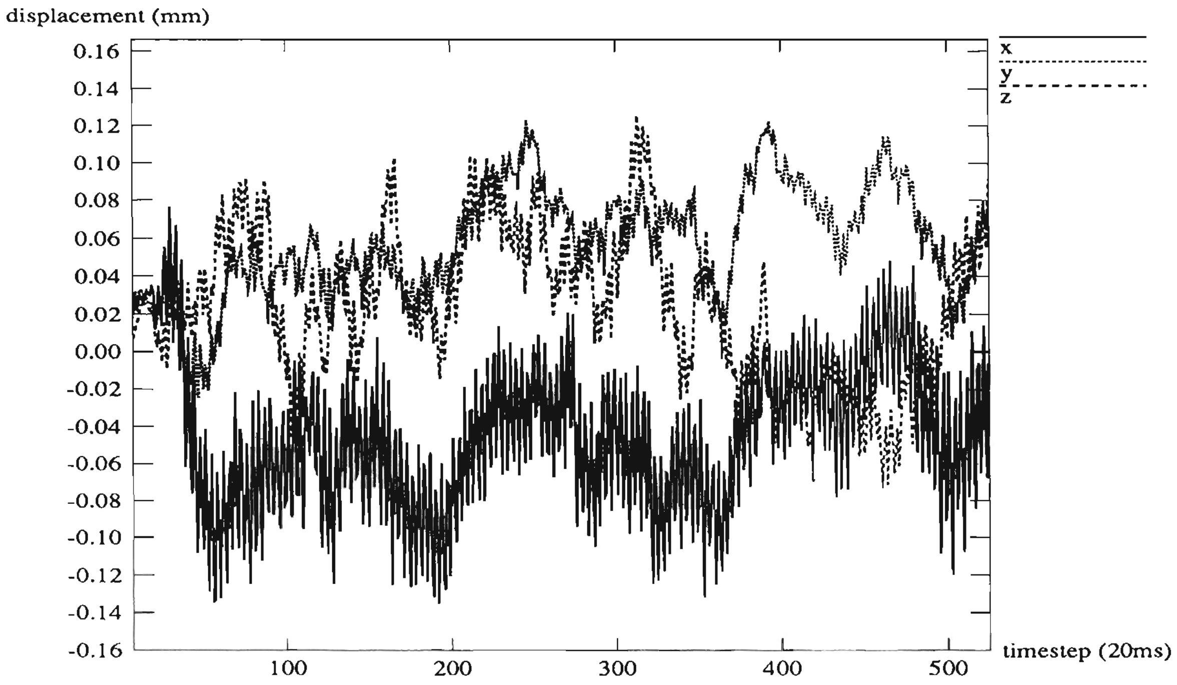


Figure 4.15: Cartesian Deflection, Off-Diagonal Term = 30,000

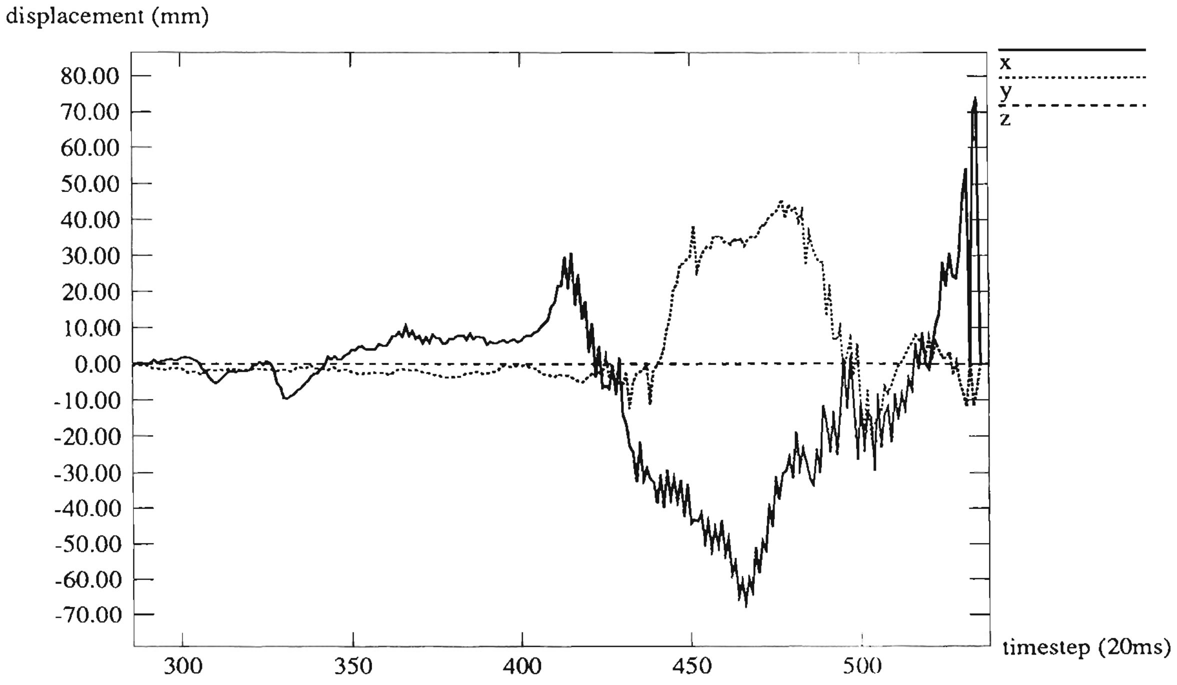


Figure 4.16: Cartesian Deflection, $l = 30m$

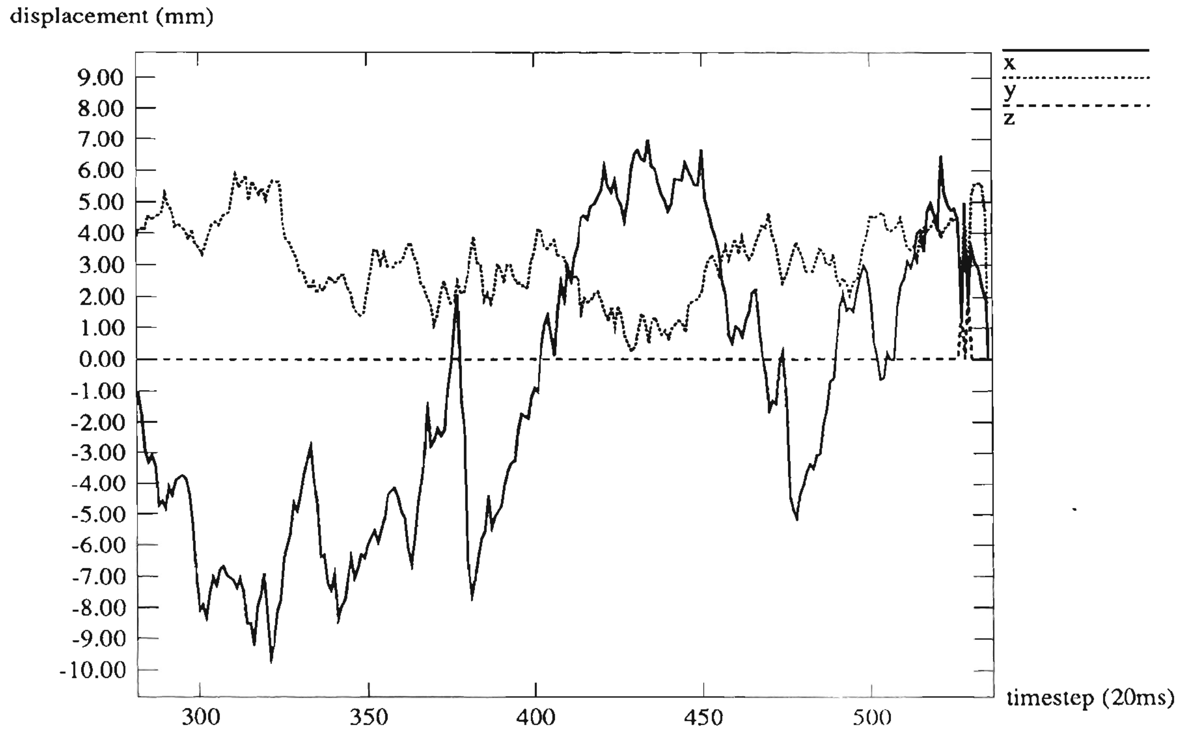


Figure 4.17: Cartesian Deflection, $l = 30m$, Reduced Gains

by the off-diagonal gain.

Figure 4.16 shows the robot controlled in a task frame translated 30m from the robot wrist in the z-direction of the tool frame. The x- and y-direction displacements vary greatly from the desired position. The z-direction, however, is relatively unaffected by the change, as there is no amplification of noise in this direction. By reducing the gains by half, the motion of the manipulator is reduced by an order of magnitude (figure 4.17).

4.7 Conclusions

Hybrid position/force control using an instrumented compliant wrist for sensory feedback is effective for controlling a robot manipulator in a partially known, unstructured environment. The impact of expected and unexpected collisions is absorbed by the wrist, which allows interaction with the environment when the robot servo interrupt is only 20ms. Mode switching (position mode for free directions, force mode for constrained directions) is not a problem with this system. Arbitrary task frames can be used with this control structure. The system remains stable using arbitrary task frames unless the frame becomes distant from the wrist (approximately 30m in the current system). When a remote frame is used, it is recommended that the system gains be reduced. The system will then remain stable, but control will be less responsive.

Improved Instrumented Compliant Wrist Design[†]

Thomas Lindsay and Richard P. Paul

Abstract

Interaction between robot and environment is an extremely important aspect of robotic research. Compliance helps reduce the impact effects of robot/environment interaction. Hybrid position/force control is important in most robotic tasks; accurate position control is needed in unconstrained directions, and accurate force control is needed in constrained directions. Force control can be more responsive with a compliant force/torque sensor, but positional accuracy is reduced with compliance. An instrumented compliant wrist device can be used to achieve both responsive force control and accurate position control.

The wrist is connected in series between the end of the robot and the tool, and is designed to partially surround the tool, thus reducing the distance between the end of the robot and the end of the tool. The wrist device uses rubber elements for compliance and damping, and a serial linkage, with potentiometers at each joint, is used for sensing the deflections produced in the wrist.

This document describes the newest version of the instrumented compliant wrist, including modifications and improvements to the wrist described in "Design of a Tool Surrounding Compliant Instrumented Wrist", available as tech report MS-CIS-91-30, GRASP LAB 258 from the University of Pennsylvania. Changes include a more protective sensing linkage structure and improved electronics. The compliance, kinematics, and accuracy of the wrist are presented. Also, software for determining the wrist transform, and plans for the wrist are given.

[†]This material is based upon work supported by the National Science Foundation under Grant No. 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.

Contents

1 Introduction **4**

2 Compliant Structure **5**

3 Linkage Kinematics **9**

4 Accuracy **12**

4.1 Positional Accuracy 13

4.2 Hysteresis Effects 14

5 Wrist Software **15**

6 Wrist Plans **17**

6.1 General 17

6.2 Compliant Structure 20

6.3 Sensing Linkage 23

6.4 Electronics 29

7 Conclusion and Future Work **33**

List of Figures

1 The Tool-Surrounding Instrumented Compliant Wrist . . . 4
2 Compliant Structure . . . 5
3 Compliant Structure Measurements . . . 6
4 Rubber Element Placement . . . 8
5 Serial Linkage Chain . . . 9
6 Kinematic Skeleton . . . 10
7 Side view of linkage structure . . . 11
8 Top view of linkage structure . . . 11
9 Sensor Data from Stationary Manipulator . . . 13
10 Sensor Data from Free Space Motion . . . 13
11 Sensor Data from Sliding Motion . . . 14
12 Top Plate . . . 18
13 Bottom Plate . . . 19
14 Compliant Structure - Exploded View . . . 20
15 Compliant Structure Piece 1 . . . 21
16 Compliant Structure Piece 2 . . . 22
17 Sensing Linkage - Exploded View . . . 23
18 Linkage Piece 1 . . . 24
19 Linkage Piece 2 . . . 25
20 Linkage Piece 3 . . . 26
21 Linkage Piece 4 . . . 27
22 Linkage Piece 5 . . . 28
23 Wrist Power and Connections . . . 30
24 Wrist Electronics . . . 31
25 Signal Conditioning Board . . . 32
26 Exploration and locomotion application (courtesy, P. Sinha) . . . 33
27 Impact wrench attached to wrist . . . 34
28 Penn Hand attached to wrist (courtesy, M. Salganicoff) . . . 34

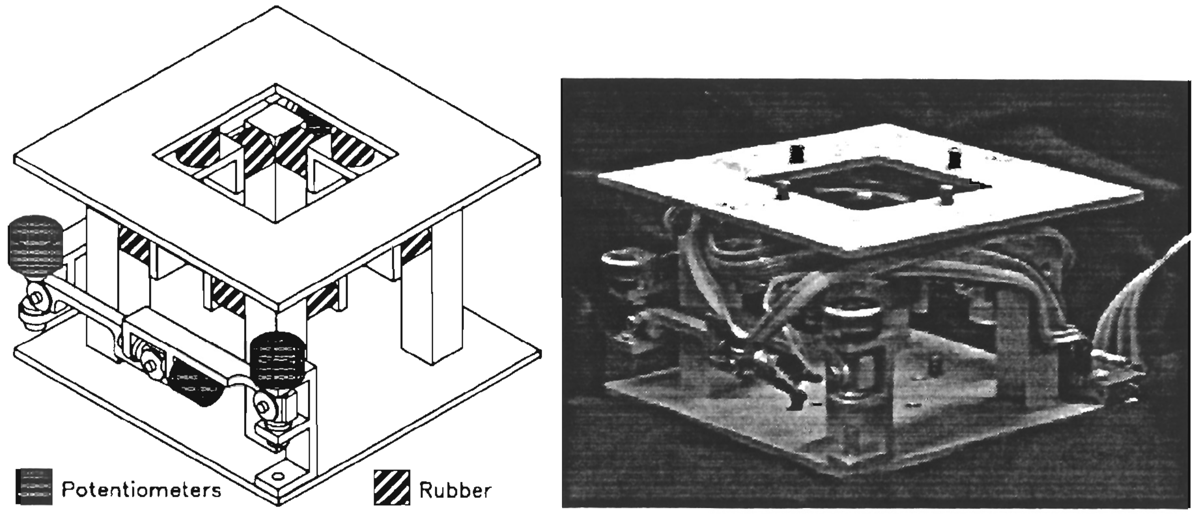


Figure 1: The Tool-Surrounding Instrumented Compliant Wrist

1 Introduction

The wrist outlined herein is a solution to a complex problem: compliance in a robot wrist is desired to reduce the effect of impacts between the robot and the environment, and to create a more responsive force control. However, a compliant wrist by itself limits the effective stiffness of the manipulator in position control, and the exact position of the end of the wrist (and thus the environment, when in contact) is lost [7]. By instrumenting the wrist as described here, both problems are overcome: active control can be used to increase the stiffness of the system, and the position transform of the wrist is sensed. Using the instrumented wrist as a compliant force/torque sensor leads to more responsive force control than with a stiff sensor [5], and more accurate position control than with a compliant wrist [8].

The wrist is overall 4.25 x 4.25 x 3.0 inches high (108 x 108 x76 mm). A 1.75 x 1.75 inch (44.5 x 44.5 mm) tool can be mounted inside the wrist to a depth of 2.5 inches (63.5 mm) maximum, depending on the desired flexure of the wrist.

This report is organized as follows: section 2 outlines the compliance of the wrist, and how it can be modified with different compliant elements, section 3 describes the sensing linkage kinematics, section 4 is a short analysis of the accuracy of the wrist, section 5 contains a simple software routine to compute the wrist transform, section 6 contains the mechanical and electrical plans for the wrist, and section 7 gives a conclusion including current research using the wrist and future work on the wrist.

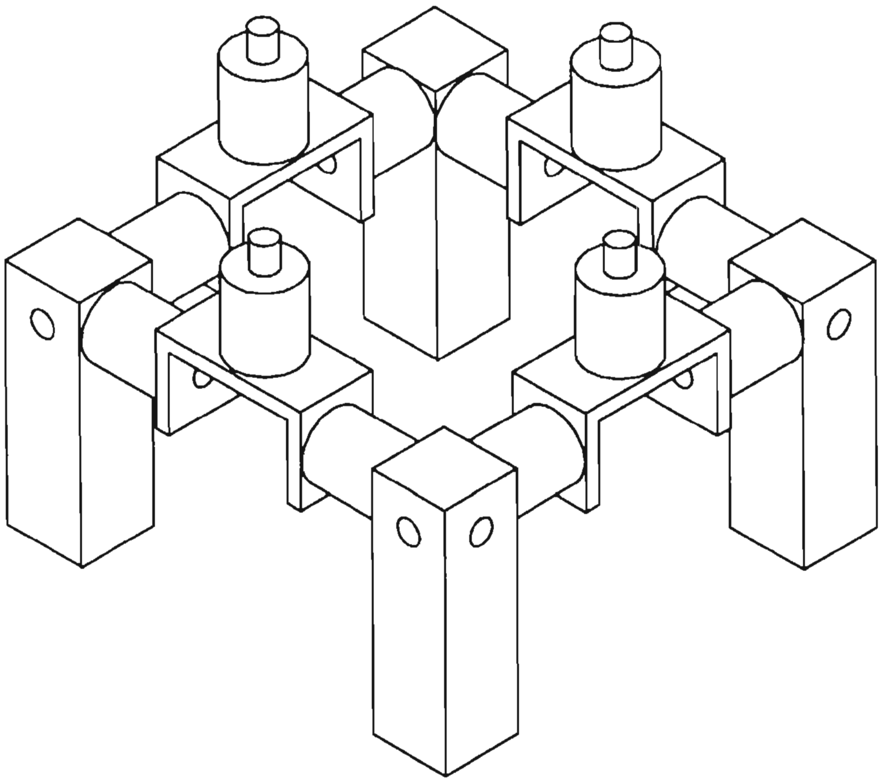


Figure 2: Compliant Structure

2 Compliant Structure

The compliant structure of the wrist is composed of 12 rubber elements, which provide compliance and a small degree of damping. Figure 2 shows the compliant structure design, with the bottom plate (attached to the robot) fixed to the four aluminum blocks at the corners, and the top plate (where the tool is attached) fixed to the four compliant elements (cylinders) at the top. The tool can be partially enclosed in the center of this structure.

The stiffness in each direction can be approximated as follows:

$$K_z = (\frac{1}{4K_a} + \frac{1}{8K_r})^{-1} \tag{1}$$

$$K_x = K_y = (\frac{1}{4K_r} + \frac{1}{4K_a + 4K_r})^{-1} \tag{2}$$

$$K_\psi = (\frac{1}{4K_r L_1^2} + \frac{1}{8K_a L_1^2})^{-1} \tag{3}$$

$$K_\phi = K_\theta = (\frac{1}{2K_a L_1^2} + \frac{1}{4K_r L_1^2 + 4K_r L_2^2})^{-1} \tag{4}$$

where K_a and K_r are the axial and shear stiffnesses of a single element, and L_1 and L_2 are shown in figure 3. This approximation uses the axial and shear stiffness of the rubber elements as supplied by the manufacturer, but ignores any bending stiffness. Age and

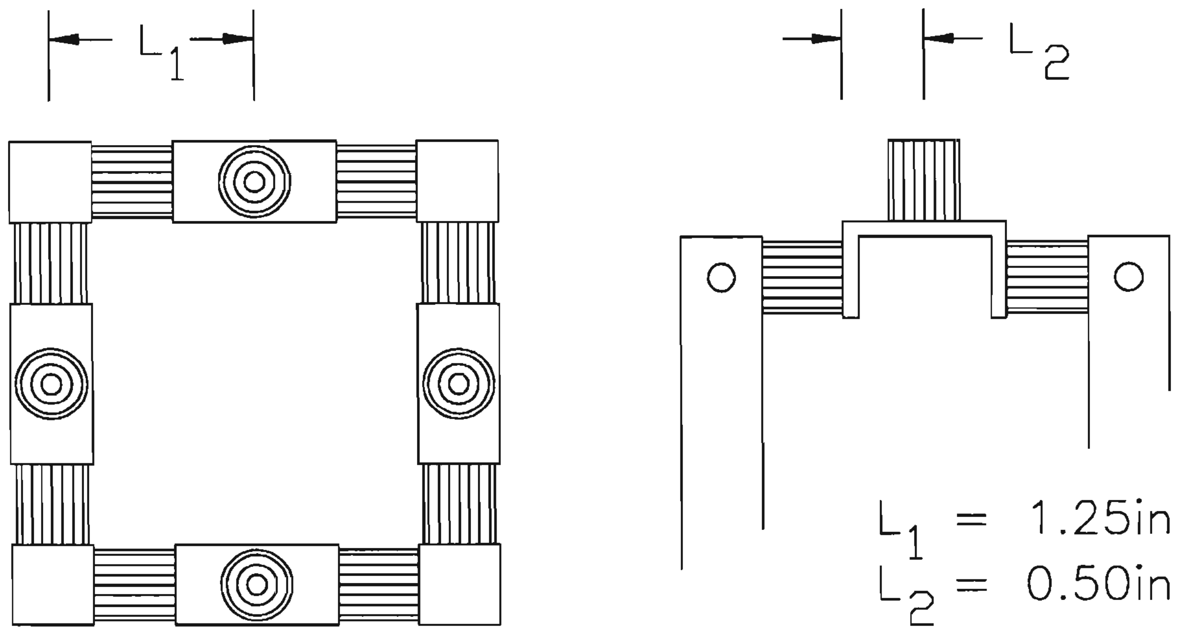


Figure 3: Compliant Structure Measurements

wear for the rubber will also change the stiffness parameters, but this effect has been ignored. For the rubber elements used, the axial stiffness K_a = 66.7 lb/in. (2.63 N/mm) and the shear stiffness K_r = 12.0 lb/in. (.472 N/mm). The approximate compliance of the wrist is tabulated below.

K_x	K_y	K_z	K_ϕ	K_θ	K_ψ
lb/in (N/mm)	lb/in (N/mm)	lb/in (N/mm)	in-lb (N-m)	in-lb (N-m)	in-lb (N-m)
41.65 (7.29)	41.65 (7.29)	70.59 (12.36)	61.37 (6.93)	61.37 (6.93)	68.81 (7.77)

The compliant structure is extremely modular. By exchanging the rubber elements for ones of different properties, the stiffness of the wrist can be changed. For example, the elements can be replaced with any of three similar elements produced by the same manufacturer, or taken away entirely. If the stiffness approximation equations shown above are broken down into individual element stiffnesses, the following equations apply (see figure 4 for element placement numbers):

$$K_z = \Big(\frac{1}{(K_{1a}+K_{2a}+K_{3a}+K_{4a})} + \frac{1}{(K_{5r}+K_{6r}+K_{7r}+K_{8r}+K_{9r}+K_{10r}+K_{11r}+K_{12r})}\Big)^{-1} \tag{5}$$

$$K_x = \Big(\frac{1}{(K_{1r}+K_{2r}+K_{3r}+K_{4r})} + \frac{1}{(K_{7a}+K_{8a}+K_{11a}+K_{12a}+K_{5r}+K_{6r}+K_{9r}+K_{10r})}\Big)^{-1} \tag{6}$$

$$K_y = \left(\frac{1}{(K_{1r}+K_{2r}+K_{3r}+K_{4r})} + \frac{1}{(K_{5a}+K_{6a}+K_{9a}+K_{10a}+K_{7r}+K_{8r}+K_{11r}+K_{12r})}\right)^{-1} \tag{7}$$

$$K_\psi = \left(\frac{1}{L_1^2(K_{1r}+K_{2r}+K_{3r}+K_{4r})} + \frac{1}{L_1^2(K_{5a}+K_{6a}+K_{7a}+K_{8a}+K_{9a}+K_{10a}+K_{11a}+K_{12a})}\right)^{-1} \tag{8}$$

$$K_\phi = \left(\frac{1}{L_1^2(K_{2a}+K_{4a})} + \frac{1}{L_1^2(K_{7r}+K_{8r}+K_{11r}+K_{12r})+L_2^2(K_{5r}+K_{6r}+K_{9r}+K_{10r})}\right)^{-1} \tag{9}$$

$$K_\theta = \left(\frac{1}{L_1^2(K_{1a}+K_{3a})} + \frac{1}{L_1^2(K_{5r}+K_{6r}+K_{9r}+K_{10r})+L_2^2(K_{7r}+K_{8r}+K_{11r}+K_{12r})}\right)^{-1} \tag{10}$$

Below is a table of axial and shear stiffness for sample compliant elements. In the current design, mount A is used for all positions on the wrist. Mount n occurs when no mount element is used for a site.

Mount [1]	K_a lb/in (N/mm)	K_r lb/in (N/mm)	comments
A	66.7 (2.63)	12.0 (.472)	mount used
B	92.3	17.6	
C	175.0	37.5	
D	228.6	50.0	
n	0.0	0.0	no mount

Below are some examples of using different elements for the compliant structure.

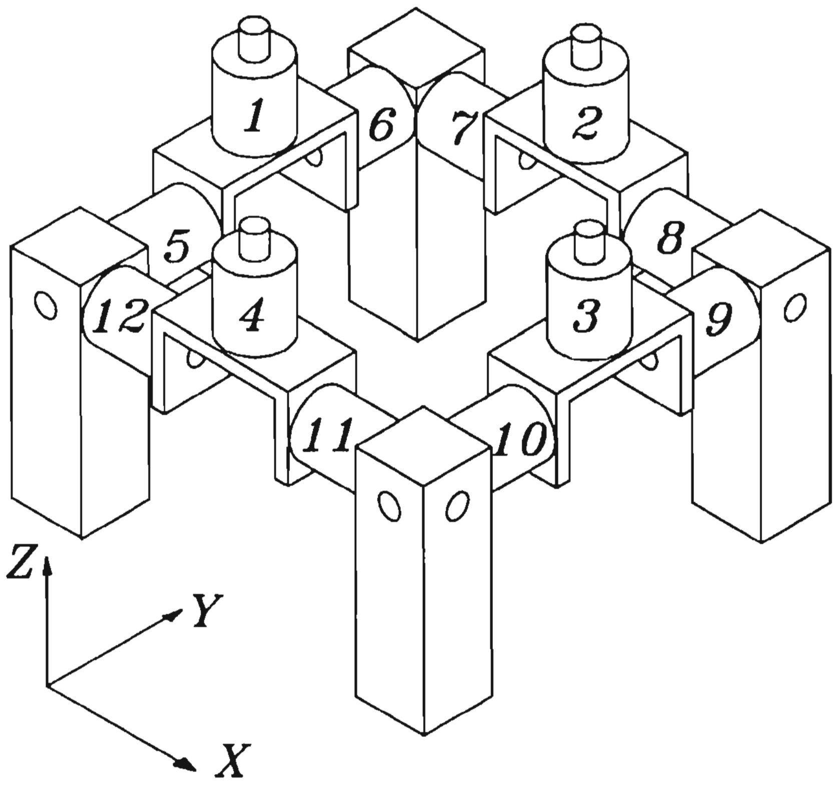


Figure 4: Rubber Element Placement

Element:	1	2	3	4	5	6	7	8	9	10	11	12
Properties:	K_x		K_y		K_z		K_θ		K_ϕ		K_ψ	
Comments	lb/in		lb/in		lb/in		lb-in		lb-in		lb-in	
current	A	A	A	A	A	A	A	A	A	A	A	A
design	41.65		41.65		70.60		61.38		61.38		68.81	
stiffest	D	D	D	D	D	D	D	D	D	D	D	D
	169.57		169.57		278.27		240.47		240.47		281.67	
most	A	A	A	A	A	n	n	A	A	n	n	A
compliant	36.78		36.78		40.68		35.99		35.99		63.56	
stiff in K_z,	A	A	A	A	D	D	D	D	D	D	D	D
K_θ, K_ϕ	46.02		46.02		160.05		132.34		132.34		73.08	
stiff in K_x,	D	D	D	D	A	A	A	A	A	A	A	A
K_y, K_ψ	122.30		122.30		86.88		77.55		77.55		227.30	
compliant	A	A	A	A	A	A	n	n	A	A	n	n
in K_x, K_θ	24.00		40.68		40.68		11.35		66.15		63.56	
even in	B	C	C	B	A	A	A	A	A	A	A	A
trans. dirs.	81.63		81.63		81.39		72.00		72.00		142.71	
even in	A	A	A	A	A	B	B	A	A	B	B	A
rot. dirs.	42.58		42.58		82.01		70.84		70.84		69.74	

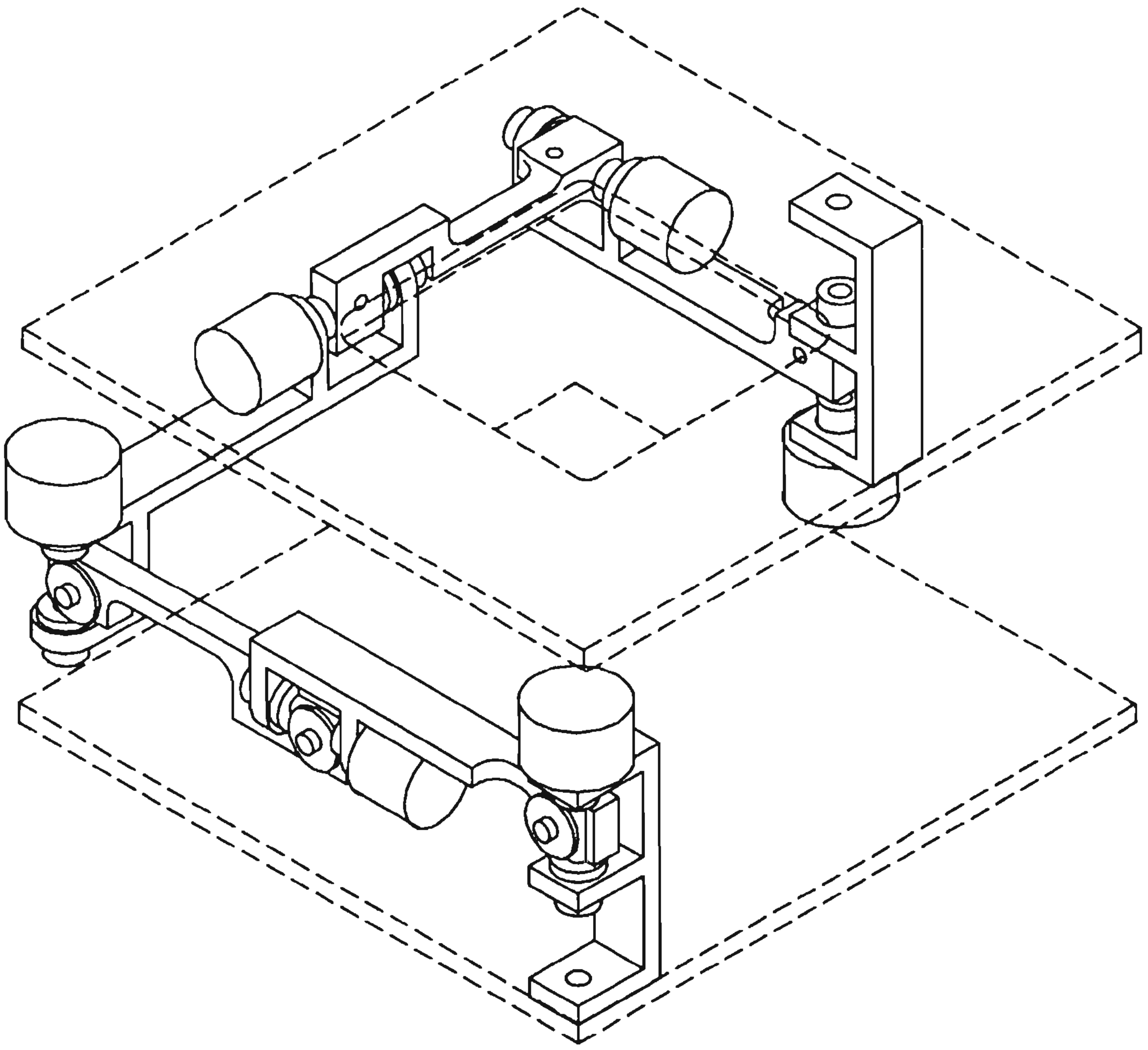


Figure 5: Serial Linkage Chain

3 Linkage Kinematics

The sensing mechanism is composed of a serial linkage chain with potentiometers at each joint (see figure 5). Voltage across the potentiometers is measured to determine the joint angles. Using a simple forward kinematic formulation, the transformation between the robot wrist and the end of the tool can be calculated. The kinematic skeleton of the wrist is shown in figure 6.

The D-H parameters for the sensing linkage mechanism, shown in figures 7 and 8 are:

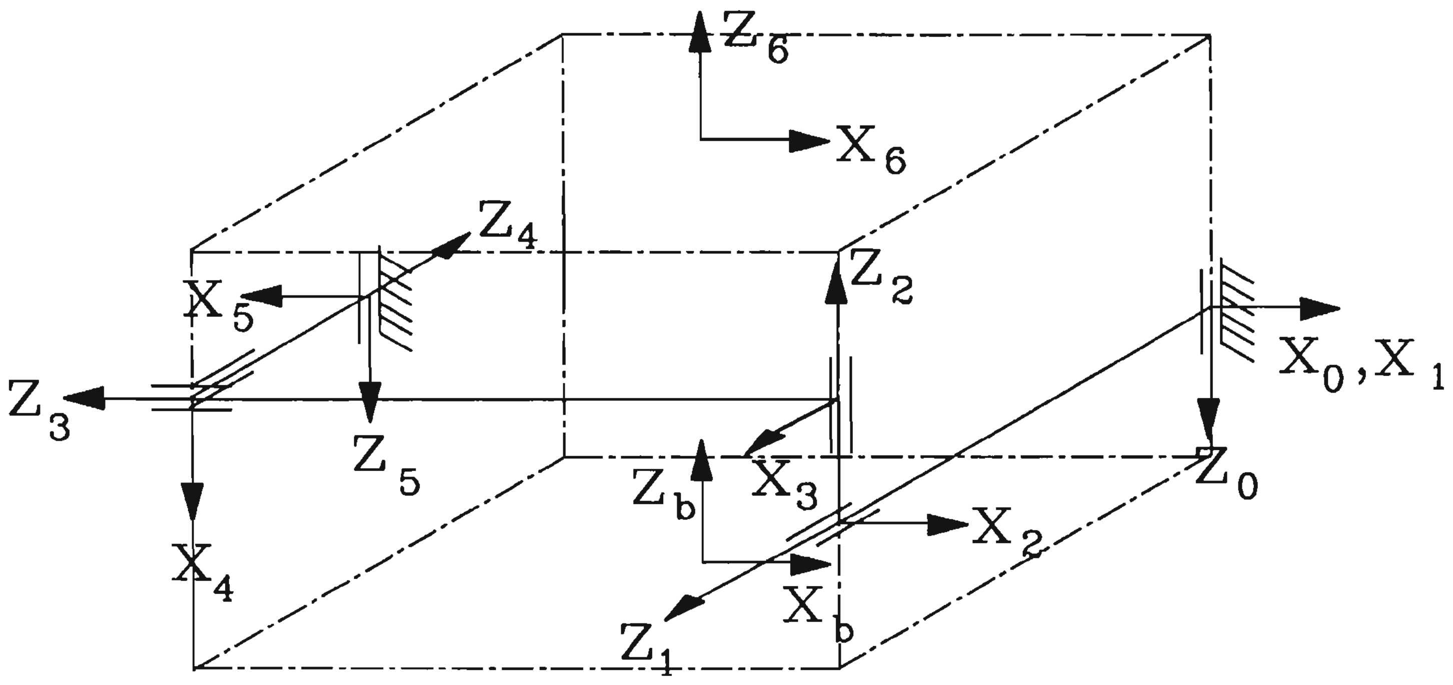


Figure 6: Kinematic Skeleton

	a	d	α	θ
joint	mm	mm	deg.	deg.
1	0	-25.00	-90	θ_1
2	0	98.82	-90	θ_2
3	0	17.86	90	-90 + θ_3
4	0	98.42	90	-90 + θ_4
5	0	49.61	90	90 + θ_5
6	49.21	-25.00	180	180 + θ_6

Also needed is the transform between the end of the robot (Frame b) and the first link frame:

$$^{b}A_0 = \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

where $l_1 = l_2 = 49.21$mm.

With this information, a transform from the end of the robot to the end of the wrist can be formed. A further transform from the end of the wrist to the end of the tool will complete the transformation from the end of the robot to the tip of the tool.

Relating the $(i-1)$th link frame to the ith link frame is a transform matrix of the form:

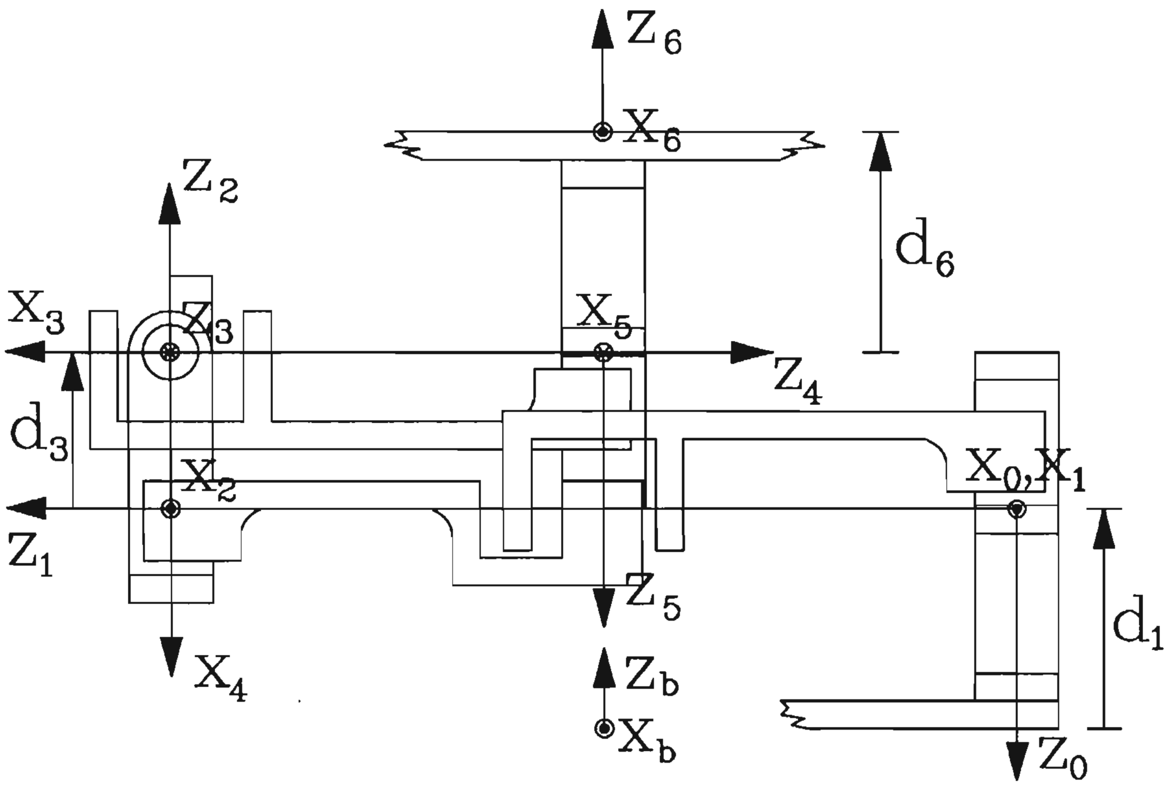


Figure 7: Side view of linkage structure

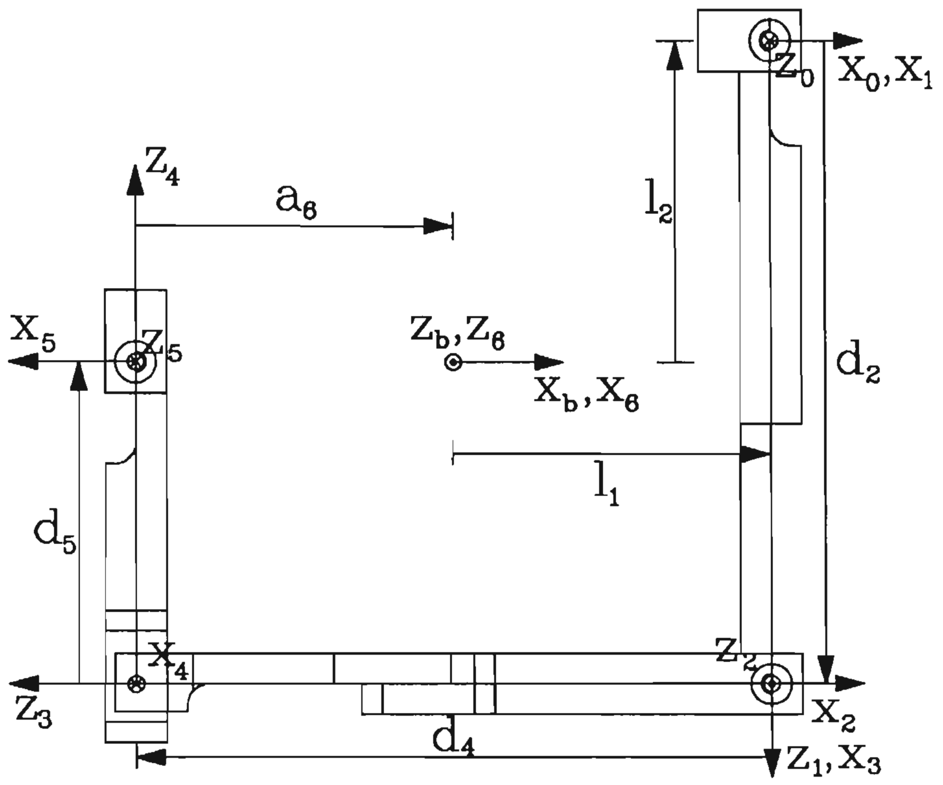


Figure 8: Top view of linkage structure

$$
{}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}
$$

The link transforms for the wrist are:

$$
{}^{0}A_1 = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}
$$

$$
{}^{1}A_2 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}
$$

$$
{}^{2}A_3 = \begin{bmatrix} \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ -\cos\theta_3 & 0 & -\sin\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}
$$

$$
{}^{3}A_4 = \begin{bmatrix} \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ -\cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}
$$

$$
{}^{4}A_5 = \begin{bmatrix} -\sin\theta_5 & 0 & \cos\theta_5 & 0 \\ \cos\theta_5 & 0 & \sin\theta_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}
$$

$$
{}^{5}A_6 = \begin{bmatrix} -\cos\theta_6 & -\sin\theta_6 & 0 & -a_6 \cos\theta_6 \\ -\sin\theta_6 & \cos\theta_6 & 0 & -a_6 \sin\theta_6 \\ 0 & 0 & -1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}
$$

Multiplying the A matrices yields ${}^{b}T_6$, the transformation of the wrist.

$$
{}^{b}T_6 =^{b} A_0 \, {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3 \, {}^{3}A_4 \, {}^{4}A_5 \, {}^{5}A_6 \tag{19}
$$

4 Accuracy

Accuracy of the wrist can be broken into two areas: positional accuracy and hysteresis effects. The positional accuracy is a function of smallest change in position that can be sensed, and sensor noise. The hysteresis effects deal with the ability of the wrist to return to a given position after it has been moved.

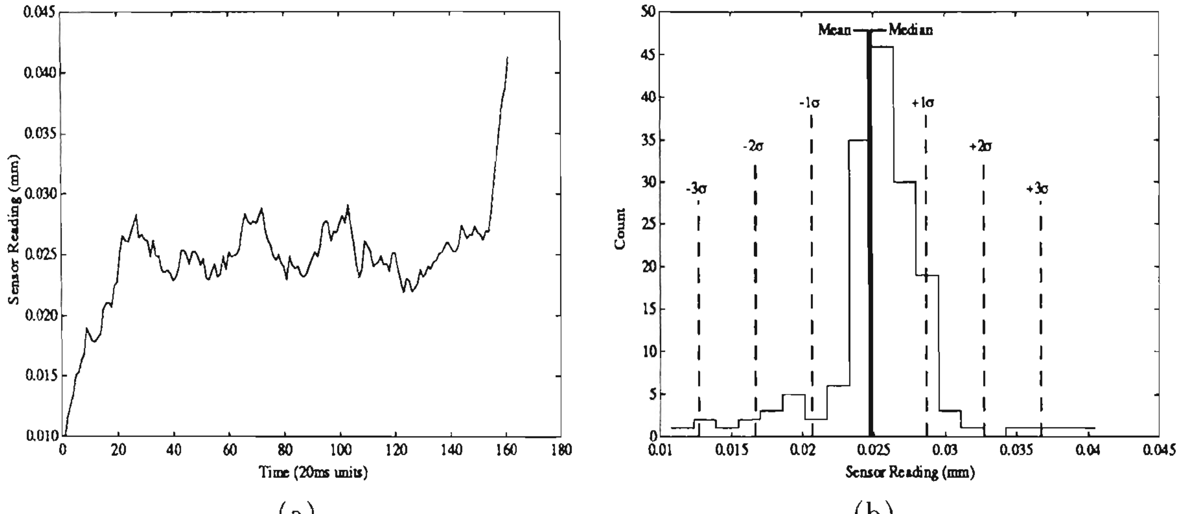


Figure 9: Sensor Data from Stationary Manipulator

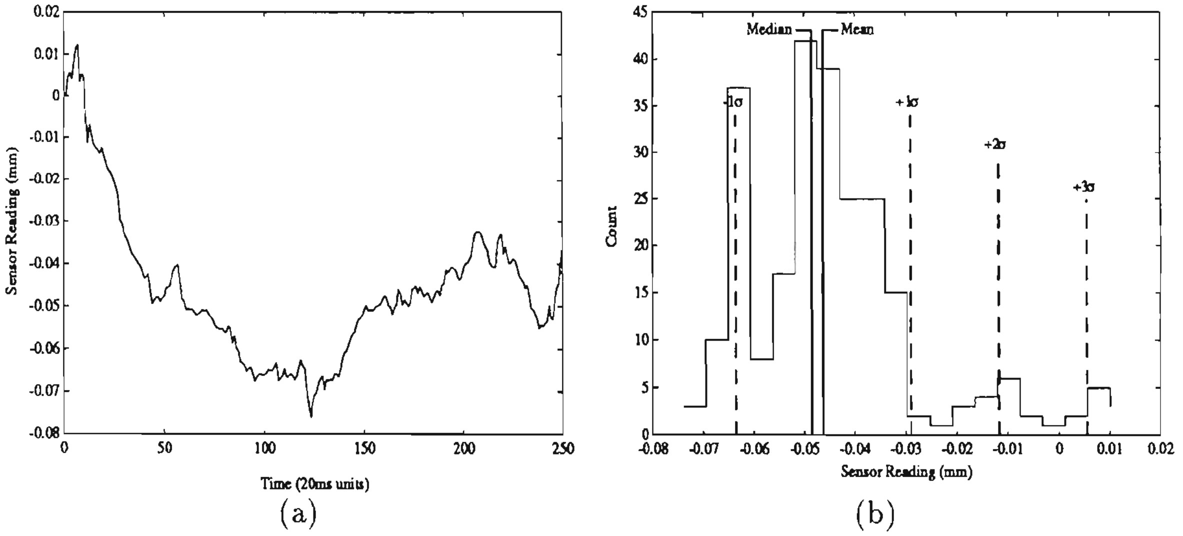


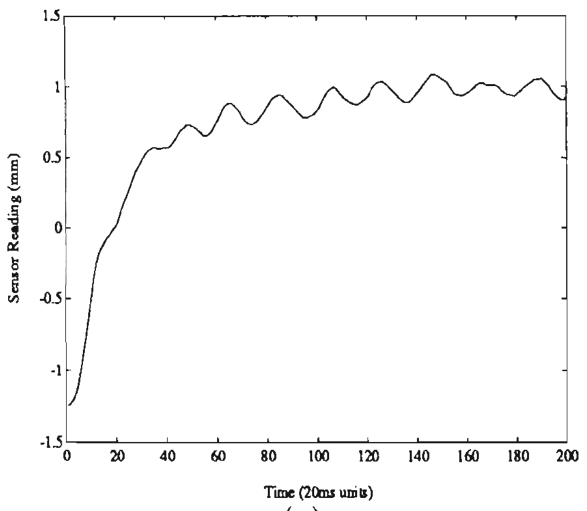
Figure 10: Sensor Data from Free Space Motion

4.1 Positional Accuracy

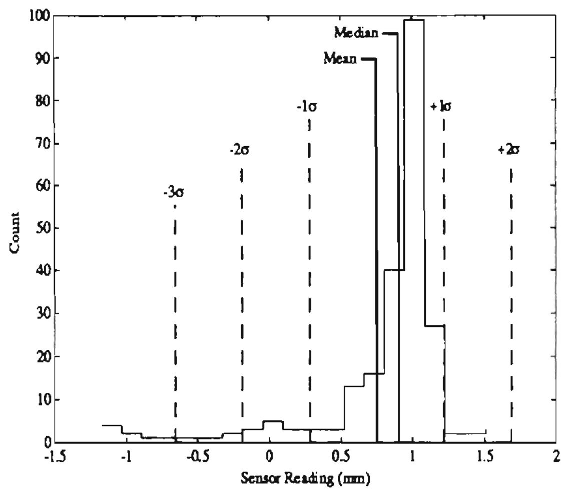
The positional accuracy of the wrist is much improved with the addition of a signal conditioner. The potentiometer voltages are differenced from a mid-range reference voltage, and amplified in the signal conditioner board, which is located in close proximity to the wrist itself in order to reduce the effects of line transmission noise. In order to assess the positional accuracy, data from a stationary wrist, free-space motion, and sliding motion (wrist sliding along a flat surface) is presented.

Figure 9 shows sensor data collected when the manipulator is stationary. The fluctuations in sensor data here are caused solely by electrical noise. The actual data is shown in figure 9(a), and the distribution of this data, in histogram format, is shown in 9(b).

Figure 10 shows sensor data from a free-space constant velocity motion, with data collected in the direction of motion. The data fluctuations here are caused by both the electrical noise, as above, and the motion vibrations of the wrist/robot system.



(a)



(b)

Figure 11: Sensor Data from Sliding Motion

Figure 11 shows sensor data from a sliding motion, while the manipulator is in contact with a surface. Data again is collected in the direction of motion. The fluctuations present in this data result from the electrical and mechanical noise, as above, as well as additional noise associated with the sliding motion. This sliding noise results from:

- Non-ideal control laws which cause the normal force with the surface to fluctuate slightly.
- Non-homogeneous surface friction.
- Coupling of orthogonal forces.

Although data for the sliding motion is highly dependent upon the control laws used for such a motion, it has been presented here as an example of the positional accuracy that may be obtained in a typical application.

Tabulated below are the statistical parameters from the three motions described above.

	mean	median	σ
no motion	0.0247	0.0249	0.0040
free space	-0.0463	-0.0485	0.0173
slide	0.7510	0.9027	0.4685

4.2 Hysteresis Effects

The non-zero mean in the no motion data is an example of the hysteresis effects (data from the wrist in motion is not expected to have a zero mean). Hysteresis effects account for a large portion of the wrist inaccuracy. Factors that contribute to the hysteresis are the natural hysteresis of the rubber compliant elements, which is so small as to be barely noticeable, and effects from friction of the potentiometers coupled with a small amount of bending in the sensing linkage structure.

Tests show that the worst-case inaccuracy due to hysteresis is approximately .6 mm (.025 in.) for translation and .0099 radians (.57 degrees) for rotation. It is clear that this far outweighs the positional inaccuracy.

5 Wrist Software

Below is a listing (in C) for a subroutine to find the wrist transform. Note that in the code, the following are defined:

$$u = {}^bA_0\,{}^0A_1\,{}^1A_2\,{}^2A_3\,{}^3A_4 \quad (20)$$
$$v = u\,{}^4A_5 \quad (21)$$
$${}^bT_6 = v\,{}^5A_6 \quad (22)$$

```
/*-----------------------------------------------------------------------*/
/* WristUpdate(car_diffs)
 *
 *  Reads current angles of wrist sensor and  computes the wrist
 *   transform, also puts the difference of cartesian deflection (from
 *   home position) in the array car_diffs[6]
*/
/*-----------------------------------------------------------------------*/

WristUpdate(car_diffs,tw)
float car_diffs[N];
struct transform tw;
{
  float   c1,c2,c3,c4,c5,c6,         /* angle cosines */
  s1,s2,s3,s4,s5,s6,             /* angle sines   */
  u11,u12,u13,u14,u21,u22,u23,u24,u31,u32,u33,u34,
  v11,v12,v13,v14,v21,v22,v23,v24,v31,v32,v33,v34;
  float ang;
  /* Link parameter values */
  float l1 = 49.21, l2 = 49.21;
  float d1 = -25.00,d2 = 98.82,d3 = 17.86,d4 = 98.42,d5=49.61,d6= -25.00;
  float a6 = 49.21;

  ang=rw_jang(0);             /* read joint angle from A/D board */
  c1 = cos(ang);
  s1 = sin(ang);

  ang=rw_jang(1);
  c2 = cos(ang);
  s2 = sin(ang);

  ang=rw_jang(2);
  c3 = cos(ang);
  s3 = sin(ang);
```

```
ang=rw_jang(3);
c4 = cos(ang);
s4 = sin(ang);

ang=rw_jang(4);
c5 = cos(ang);
s5 = sin(ang);

ang=rw_jang(5);
c6 = cos(ang);
s6 = sin(ang);


/* u = A1 * A2 * A3 * A4 */
u11 = c1 * c2 * s3 * s4 - s1 * c3 * s4 + c1 * s2 * c4;
u12 = -c1 * c2 * c3 - s1 * s3;
u13 = -c1 * c2 * s3 * c4 + s1 * c3 * c4 + c1 * s2 * s4;
u14 = -d4 * c1 * c2 * c3 - d4 * s1 * s3
- d3 * c1 * s2 - d2 * s1 + l1;

u21 = -s1 * c2 * s3 * s4 - c1 * c3 * s4 - s1 * s2 * c4;
u22 = s1 * c2 * c3 - c1 * s3;
u23 = s1 * c2 * s3 * c4 + c1 * c3 * c4 - s1 * s2 * s4;
u24 = d4 * s1 * c2 * c3 - d4 * c1 * s3
+ d3 * s1 * s2 - d2 * c1 + l2;

u31 = s2 * s3 * s4 - c2 * c4;
u32 = -s2 * c3;
u33 = -s2 * s3 * c4 - c2 * s4;
u34 = -d4 * s2 * c3 + d3 * c2 - d1;

/* v = u * A5 */
v11 = -u11 * s5 + u12 * c5;
v12 = u13;
v13 = u11 * c5 + u12 * s5;
v14 = d5 * u13 + u14;

v21 = -u21 * s5 + u22 * c5;
v22 = u23;
v23 = u21 * c5 + u22 * s5;
v24 = d5 * u23 + u24;

v31 = -u31 * s5 + u32 * c5;
v32 = u33;
v33 = u31 * c5 + u32 * s5;
v34 = d5 * u33 + u34;
```

```
  /* Wrist transform */
  tw.n.x = -v11 * c6 - v12 * s6;
  tw.o.x = -v11 * s6 + v12 * c6;
  tw.a.x = -v13;
  tw.p.x = -a6 * v11 * c6 - a6 * v12 * s6 + d6 * v13 + v14;

  tw.n.y = -v21 * c6 - v22 * s6;
  tw.o.y = -v21 * s6 + v22 * c6;
  tw.a.y = -v23;
  tw.p.y = -a6 * v21 * c6 - a6 * v22 * s6 + d6 * v23 + v24;

  tw.n.z = -v31 * c6 - v32 * s6;
  tw.o.z = -v31 * s6 + v32 * c6;
  tw.a.z = -v33;
  tw.p.z = -a6 * v31 * c6 - a6 * v32 * s6 + d6 * v33 + v34;

  /* Compute roll, pitch, and yaw angles from wrist transform */
  noatorpy(&car_diffs[5],&car_diffs[4],&car_diffs[3],&tw);
  car_diffs[0] = tw.p.x;
  car_diffs[1] = tw.p.y;
  car_diffs[2] = tw.p.z - 67.86;  /* total wrist thickness */

}

/*------------------------------------------------------------------*/
```

6 Wrist Plans

6.1 General

The compliant structure and the sensing linkage are sandwiched between the top and bottom plate. The compliant structure is connected to the bottom plate with four 8-32 x 1/2" countersunk machine screws, and to the top plate by the compliant elements. The sensing linkage is attached to the top and bottom plates by two 8-32 x 1/8" countersunk machine screws. The wrist is connected to the robot via a quick mount mechanism (Lord Corporation, not shown), which bolts into the four 8-32 threaded holes in the bottom plate. A 20-pin connector is also attached to the bottom plate.

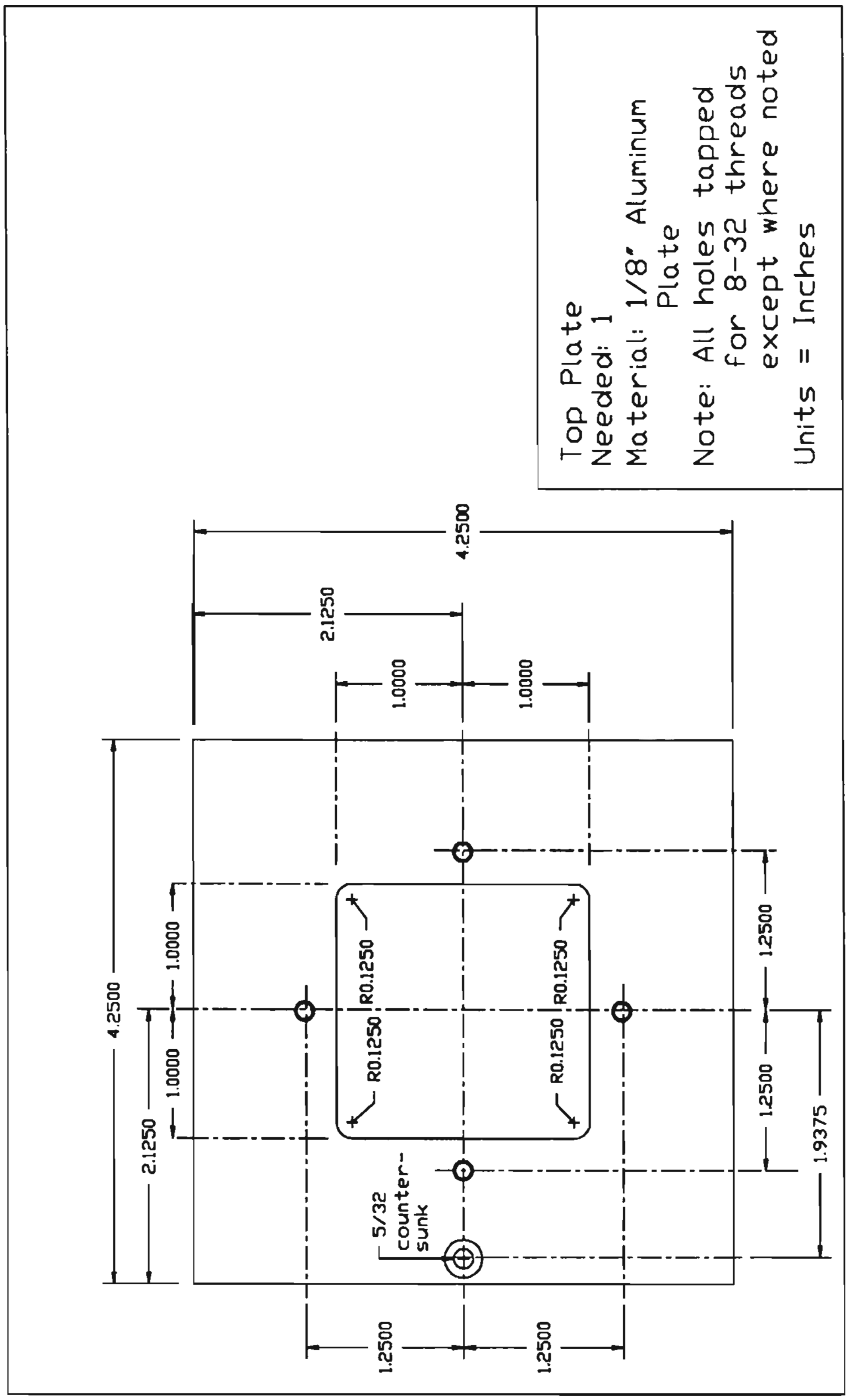
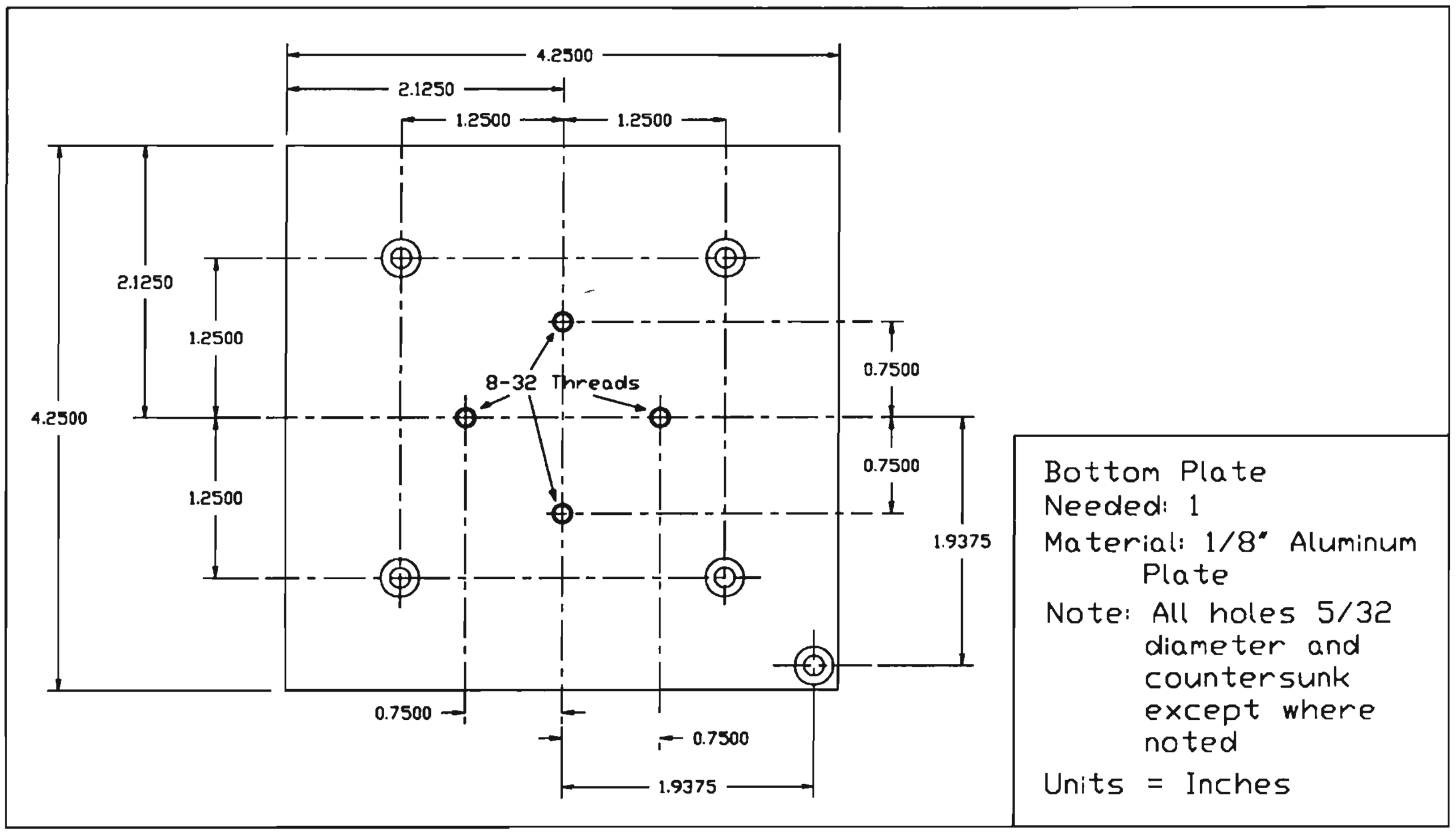


Figure 12: Top Plate



Bottom Plate
Needed: 1
Material: 1/8" Aluminum Plate
Note: All holes 5/32 diameter and countersunk except where noted
Units = Inches

Figure 13: Bottom Plate

6.2 Compliant Structure

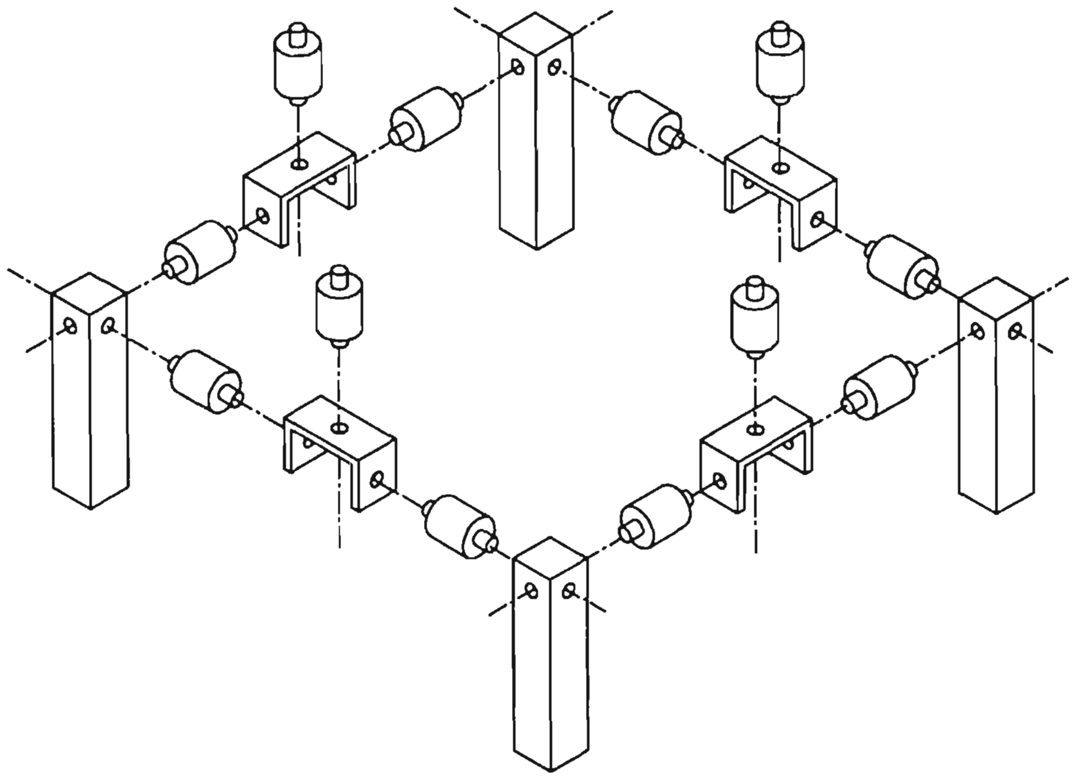


Figure 14: Compliant Structure - Exploded View

The 12 compliant elements are part number 10Z2-302A from Stock Drive Products[1]. These elements have 8-32 x 3/8" threaded studs. Four of these must have one stud shortened to 3/16", one each attached to compliant structure piece 2. All elements connected to compliant structure piece 1 are attached with 8-32 hex nuts.

[1]New Hyde Park, NY, (516) 328-0200. Stiffer elements are part numbers 10Z2-302B, 10Z2-302C, and 10Z2-302D.

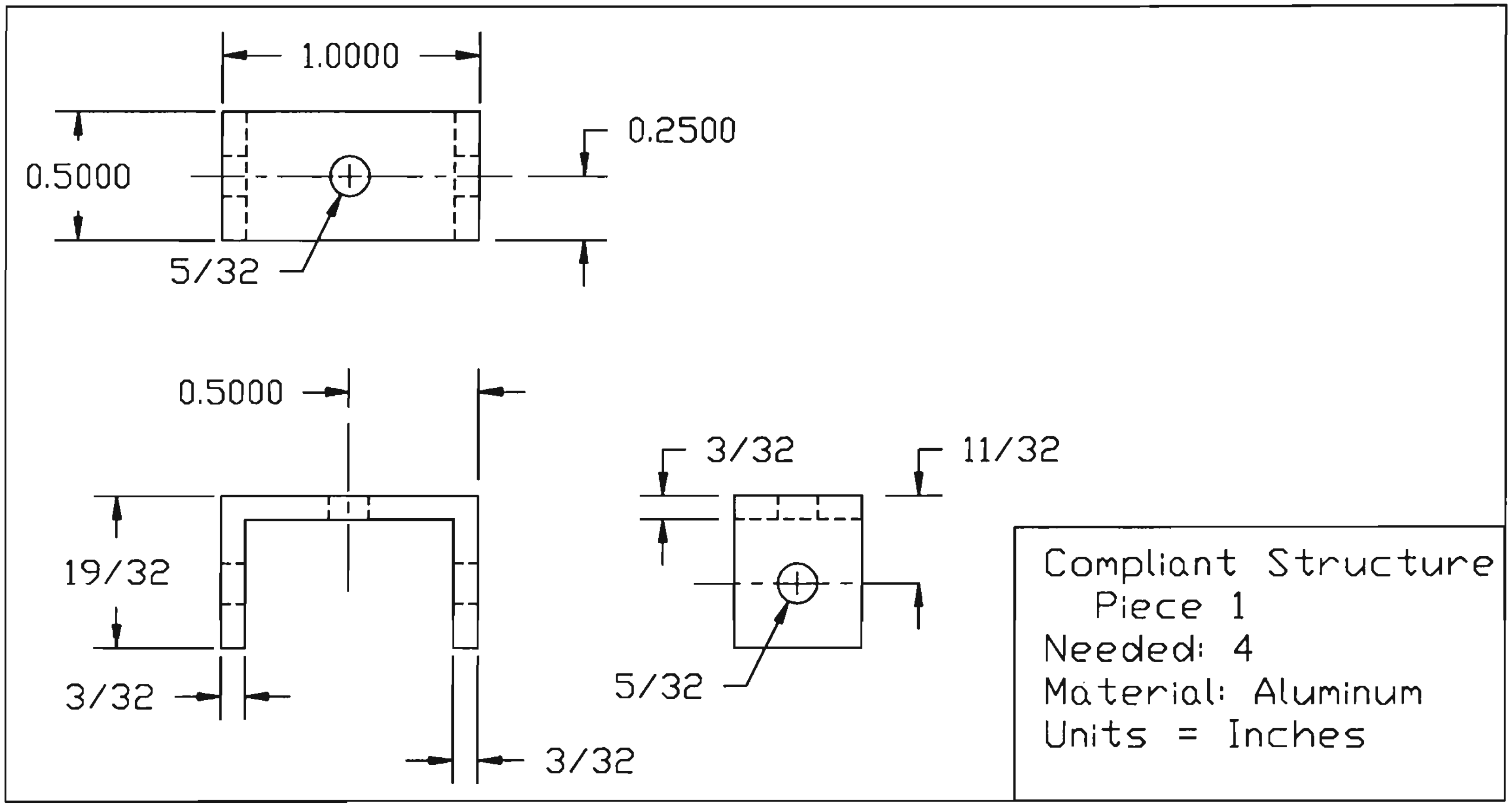


Figure 15: Compliant Structure Piece 1

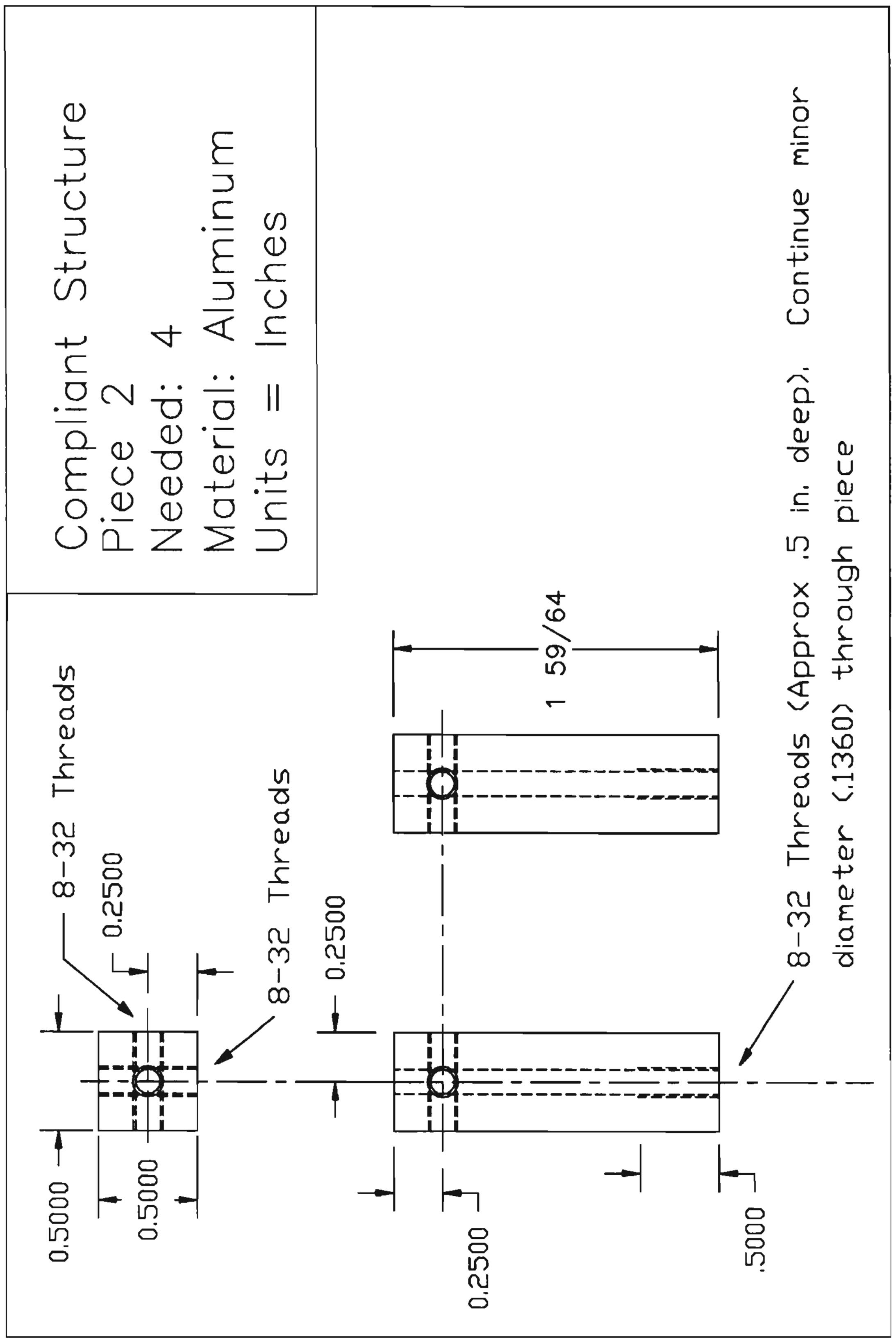


Figure 16: Compliant Structure Piece 2

6.3 Sensing Linkage

Potentiometers used at joints 0, 2, and 5 are part number 381 N 1000 S; joints 1, 3, and 4 are part number RV6 NAYSD 10 2 A from Clarostat Mfg. Co., Inc.[2]. Potentiometers are attached to linkage pieces with 4-40 x 1/4" hex head machine screws and 4-40 flat washers. Each potentiometer has an additional shaft bearing, part number K-FBB-2/4 from Small Parts, Inc.[3]

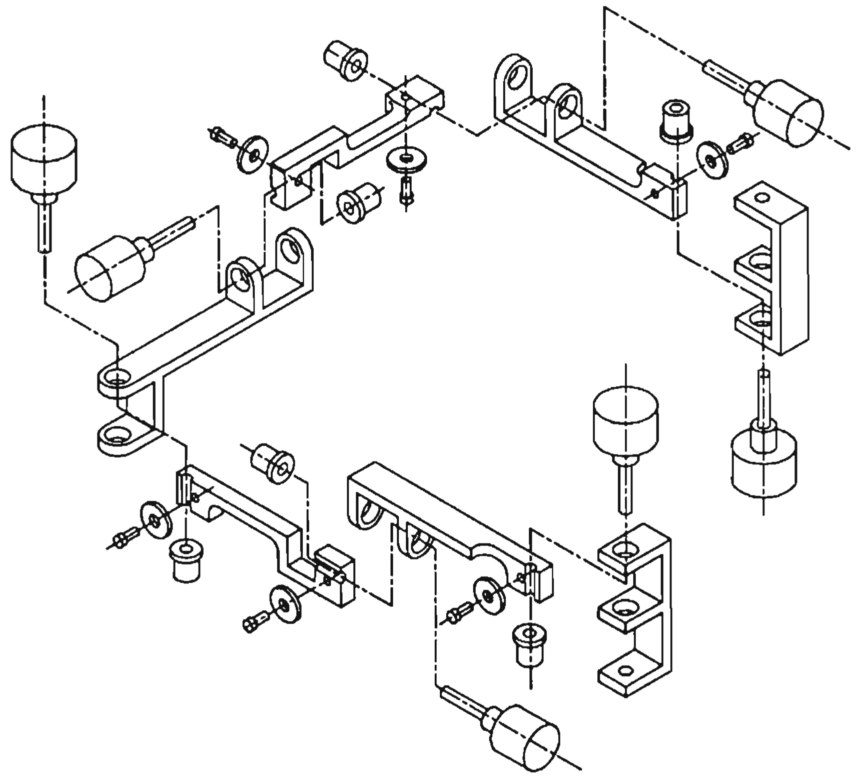


Figure 17: Sensing Linkage - Exploded View

[2]Dover, NH, (800) 872-0042.

[3]Miami Lakes, FL, (305) 557-8222.

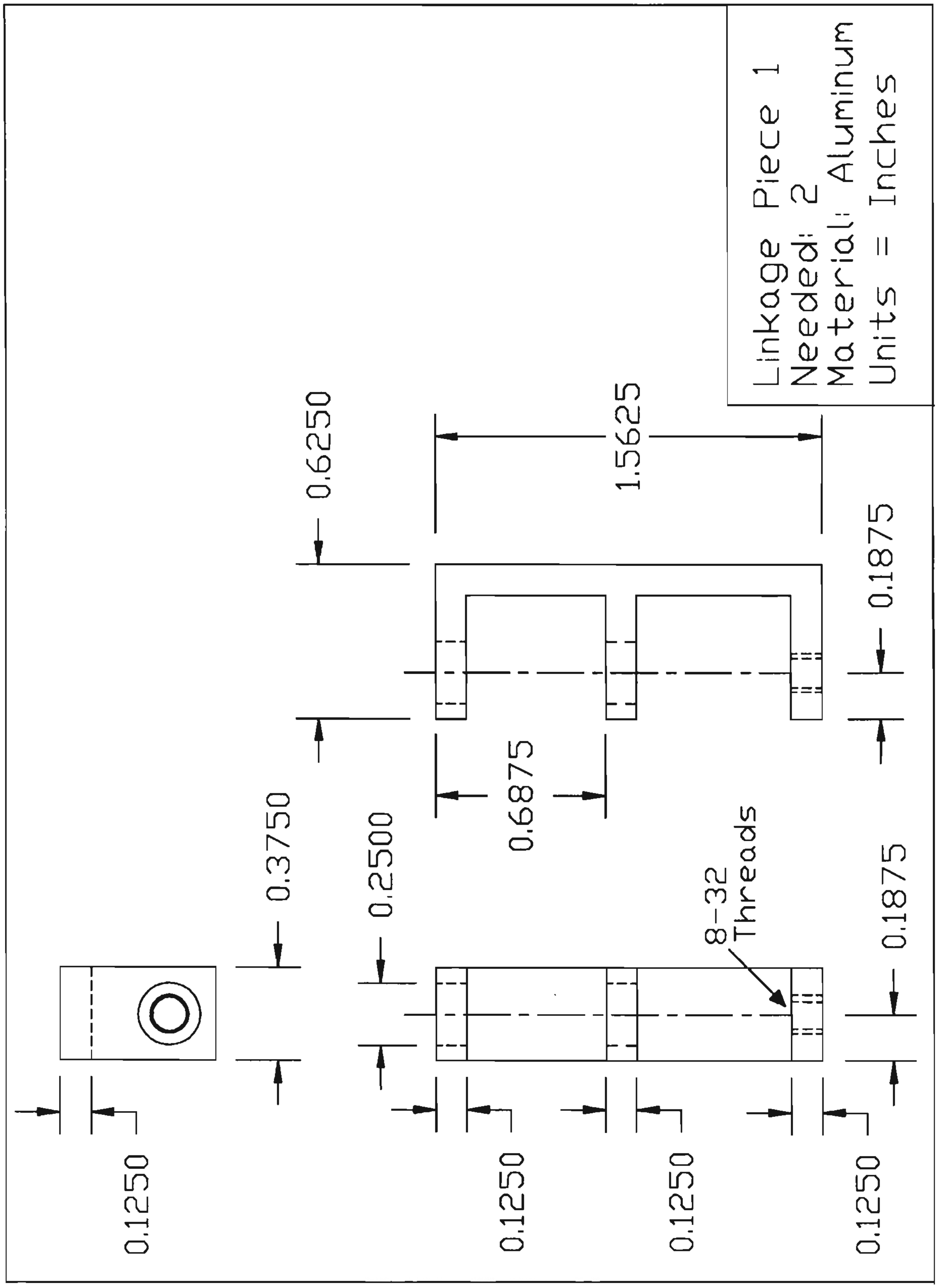


Figure 18: Linkage Piece 1

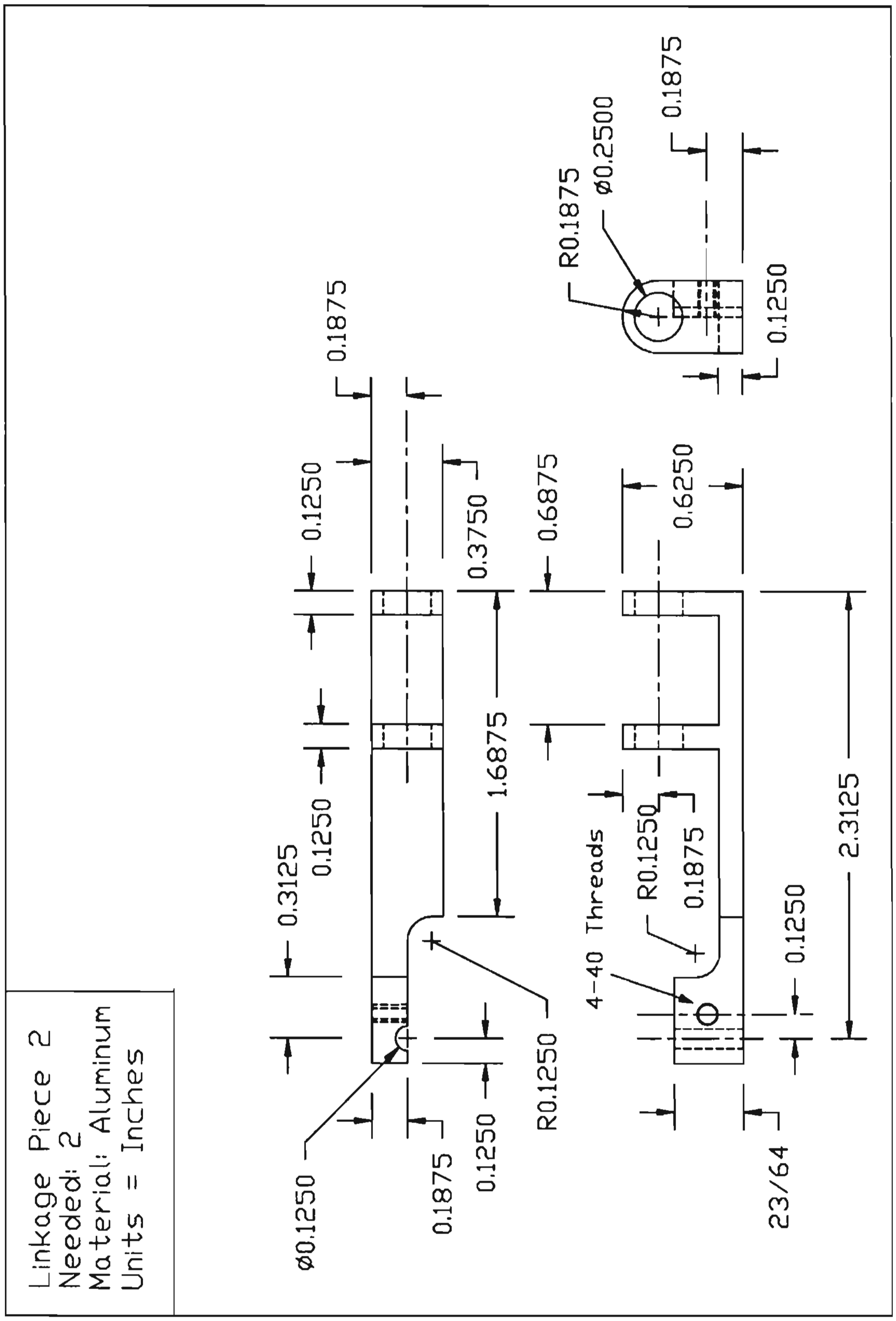


Figure 19: Linkage Piece 2

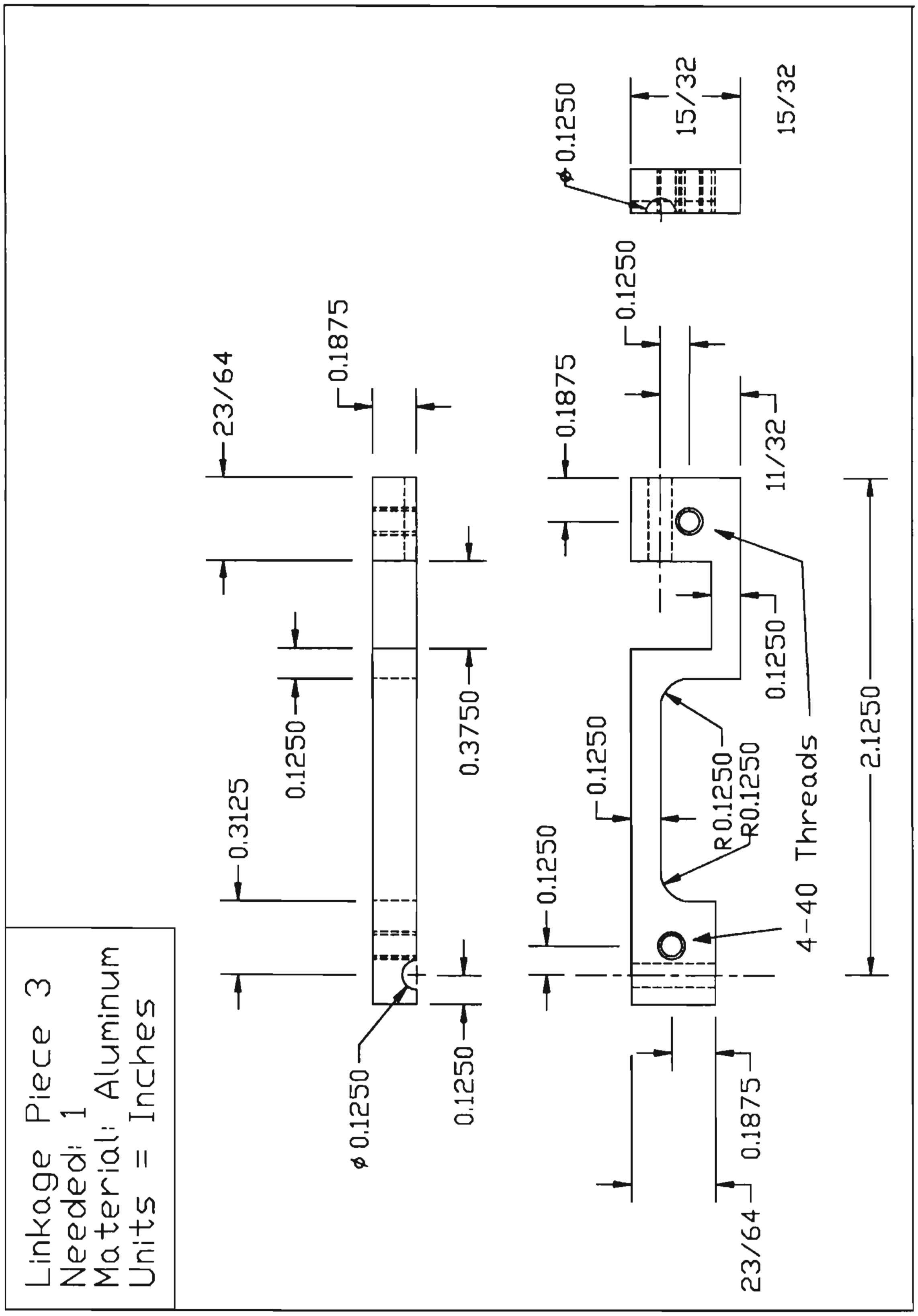


Figure 20: Linkage Piece 3

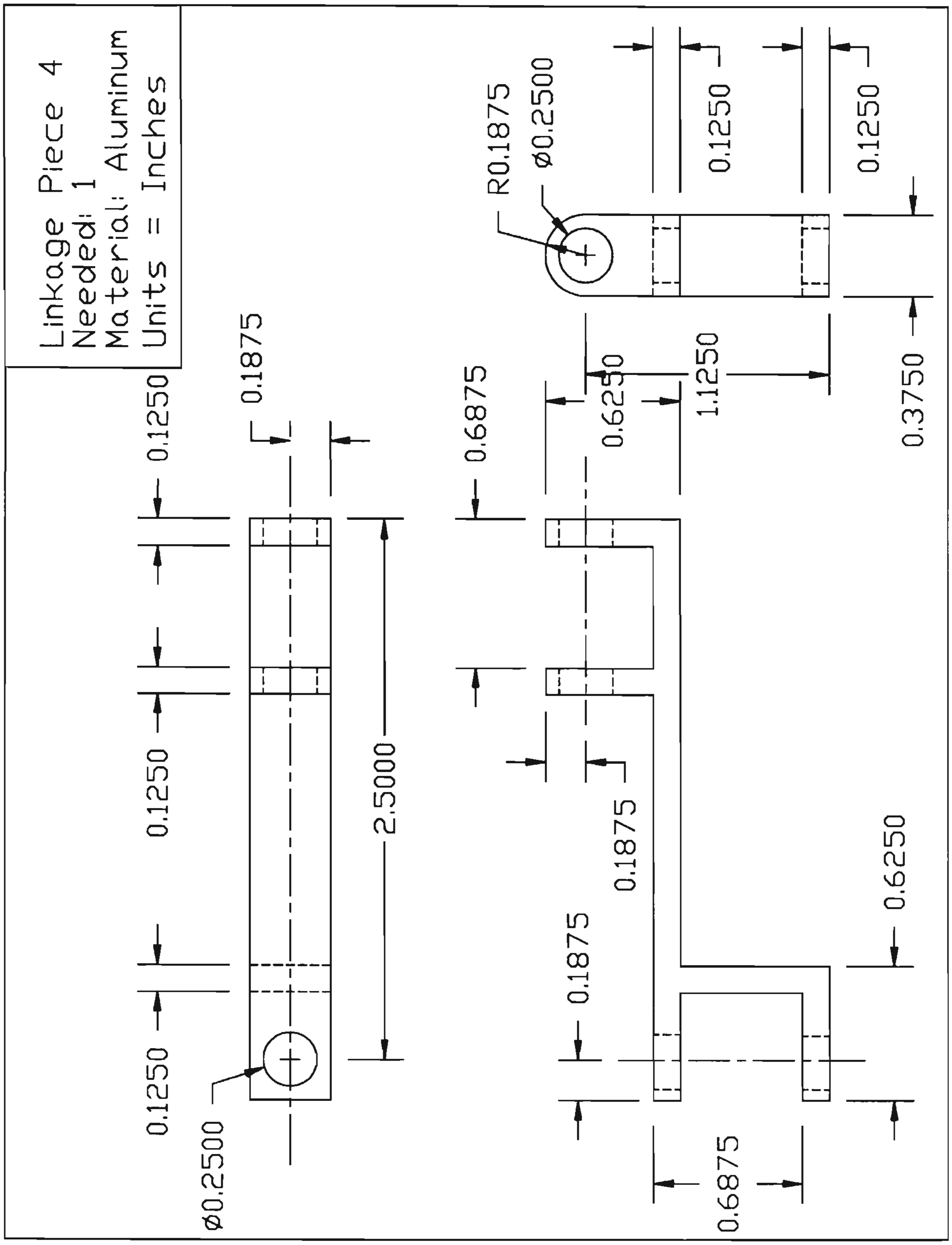


Figure 21: Linkage Piece 4

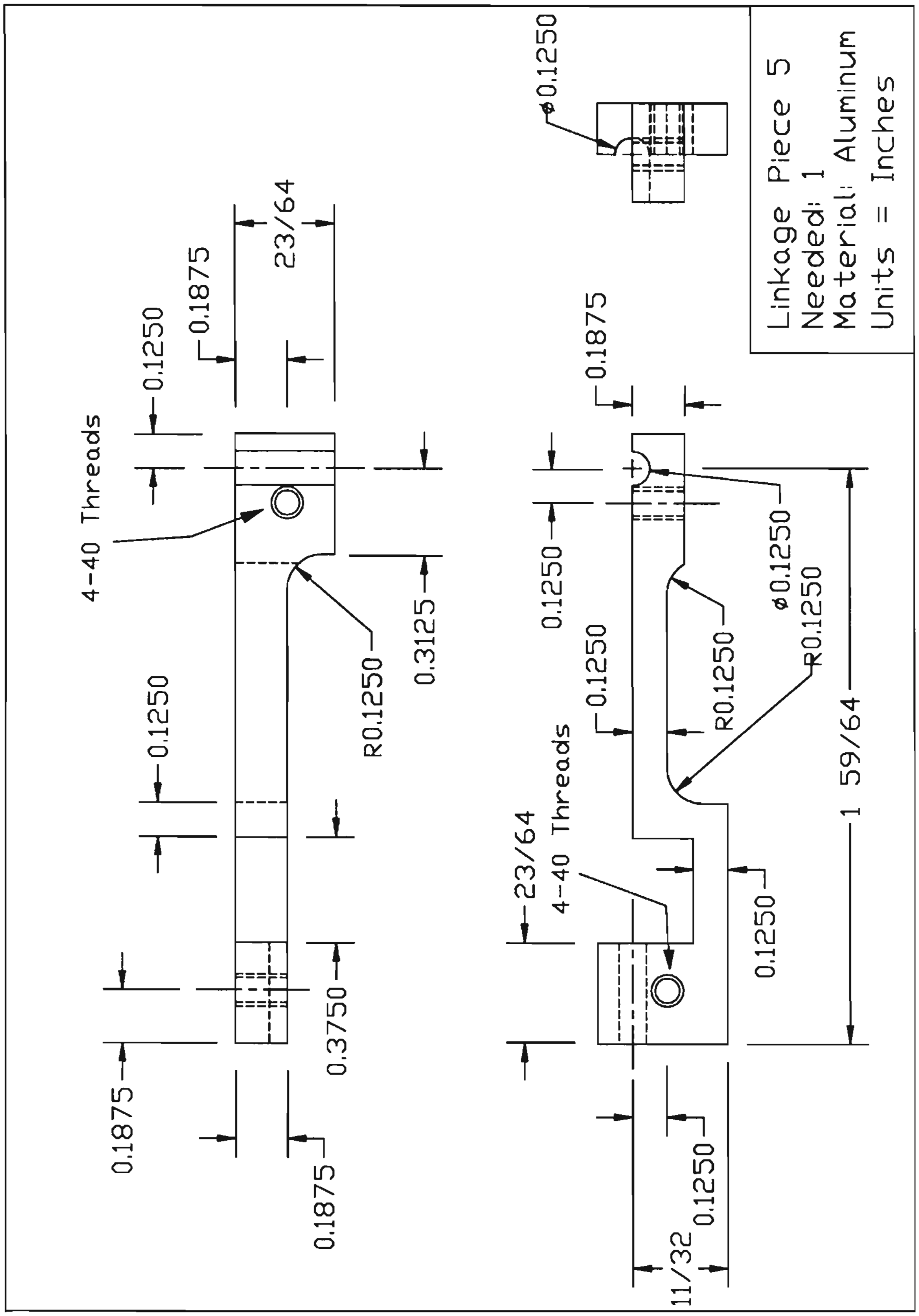


Figure 22: Linkage Piece 5

6.4 Electronics

WRIST CONNECTOR JW1			
SIGNAL NAME	PIN		SIGNAL NAME
P0G	1	2	P0T
P1G	3	4	P1T
P2G	5	6	P2T
P3G	7	8	P3T
P4G	9	10	P4T
P5G	11	12	P5T
P0R	13	14	P1R
P2R	15	16	P3R
P4R	17	18	P5R
WGND	19	20	WGND

JA1					
BOARD	SIGNAL	PIN	PIN	SIGNAL	BOARD
SGND	P0G	1	2	P0T	U1-5
SGND	P1G	3	4	P1T	U2-5
SGND	P2G	5	6	P2T	U3-5
SGND	P3G	7	8	P3T	U4-5
SGND	P4G	9	10	P4T	U5-5
SGND	P5G	11	12	P5T	U6-5
U8-7	P0R	13	14	P1R	U8-7
U8-7	P2R	15	16	P3R	U8-1
U8-1	P4R	17	18	P5R	U8-1
JA2-17	WGND	19	20	WGND	JA2-17

JA2					
BOARD	SIGNAL	PIN	PIN	SIGNAL	BOARD
U1-1	CH0	1	2	RET0	SGND
U2-1	CH1	3	4	RET1	SGND
U3-1	CH2	5	6	RET2	SGND
U4-1	CH3	7	8	RET3	SGND
U5-1	CH4	9	10	RET4	SGND
U6-1	CH5	11	12	RET5	SGND
GND	GND	13	14	GND	GND
-15v	-15v	15	16	-15v	-15v
JA1-19,20	AGND	17	18	GND	GND
+15v	+15v	19	20	+15v	+15v

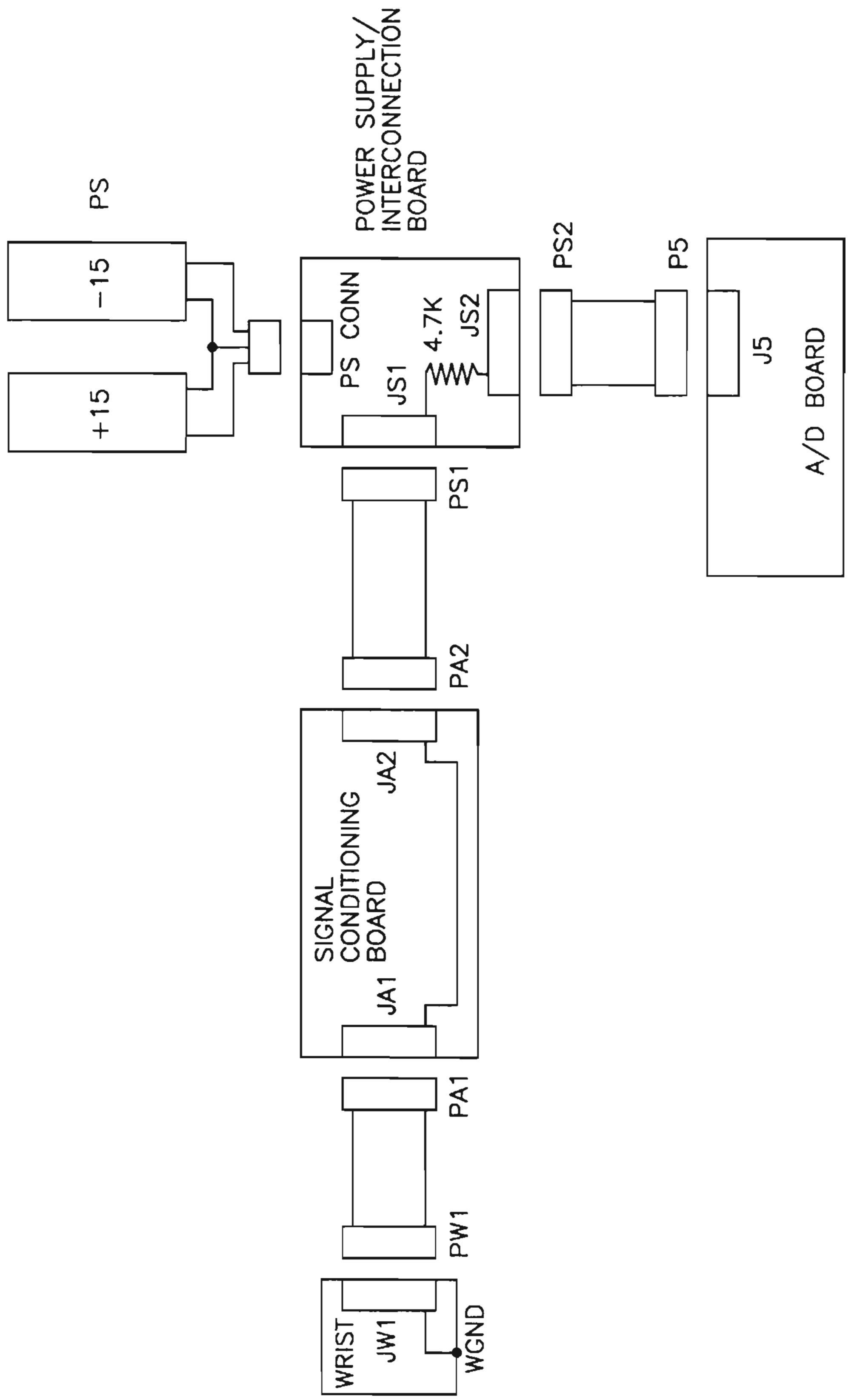


Figure 23: Wrist Power and Connections

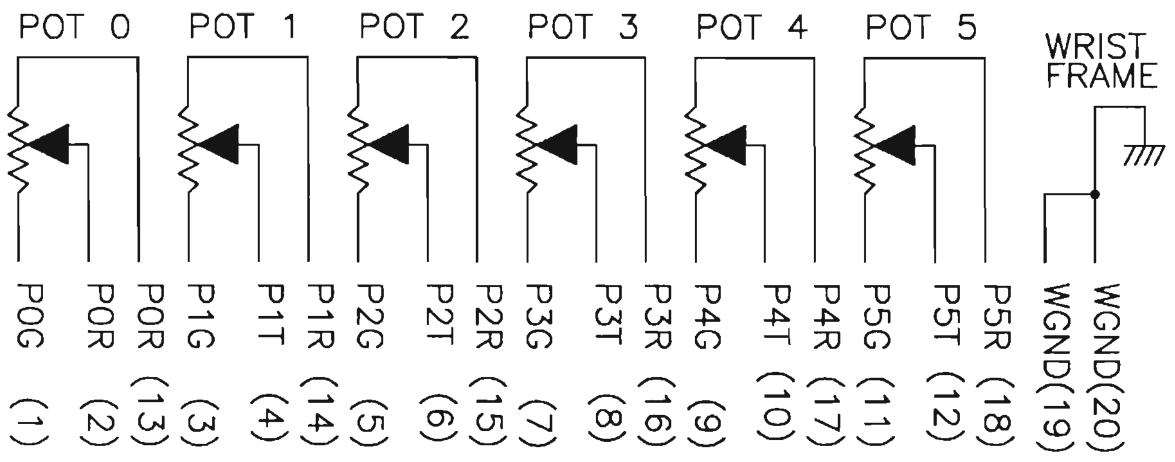


Figure 24: Wrist Electronics

JS1					
BOARD	SIGNAL	PIN	PIN	SIGNAL	BOARD
JS2-1	CH0	1	2	SGND	JS2-18
JS2-3	CH1	3	4	SGND	JS2-18
JS2-5	CH2	5	6	SGND	JS2-18
JS2-7	CH3	7	8	SGND	JS2-18
JS2-9	CH4	9	10	SGND	JS2-18
JS2-11	CH5	11	12	SGND	JS2-18
PS CONN	GND	13	14	GND	PS CONN
PS CONN	-15v	15	16	-15v	PS CONN
JS2-17	AGND	17	18	GND	PS CONN
PS CONN	+15v	19	20	+15v	PS CONN

JS2					
BOARD	SIGNAL	PIN	PIN	SIGNAL	BOARD
JS1-1	CH0	1	2	CH8	JS2-17
JS1-3	CH1	3	4	CH9	JS2-17
JS1-5	CH2	5	6	CH10	JS2-17
JS1-7	CH3	7	8	CH11	JS2-17
JS1-9	CH4	9	10	CH12	JS2-17
JS1-11	CH5	11	12	CH13	JS2-17
JS2-17	CH6	13	14	CH14	JS2-17
JS2-17	CH7	15	16	CH15	JS2-17
JS1-17	AGND	17	18	AMP LO	JS1-2,4,6,8,10,12
N/C	+12v OUT	19	20	-12v OUT	N/C

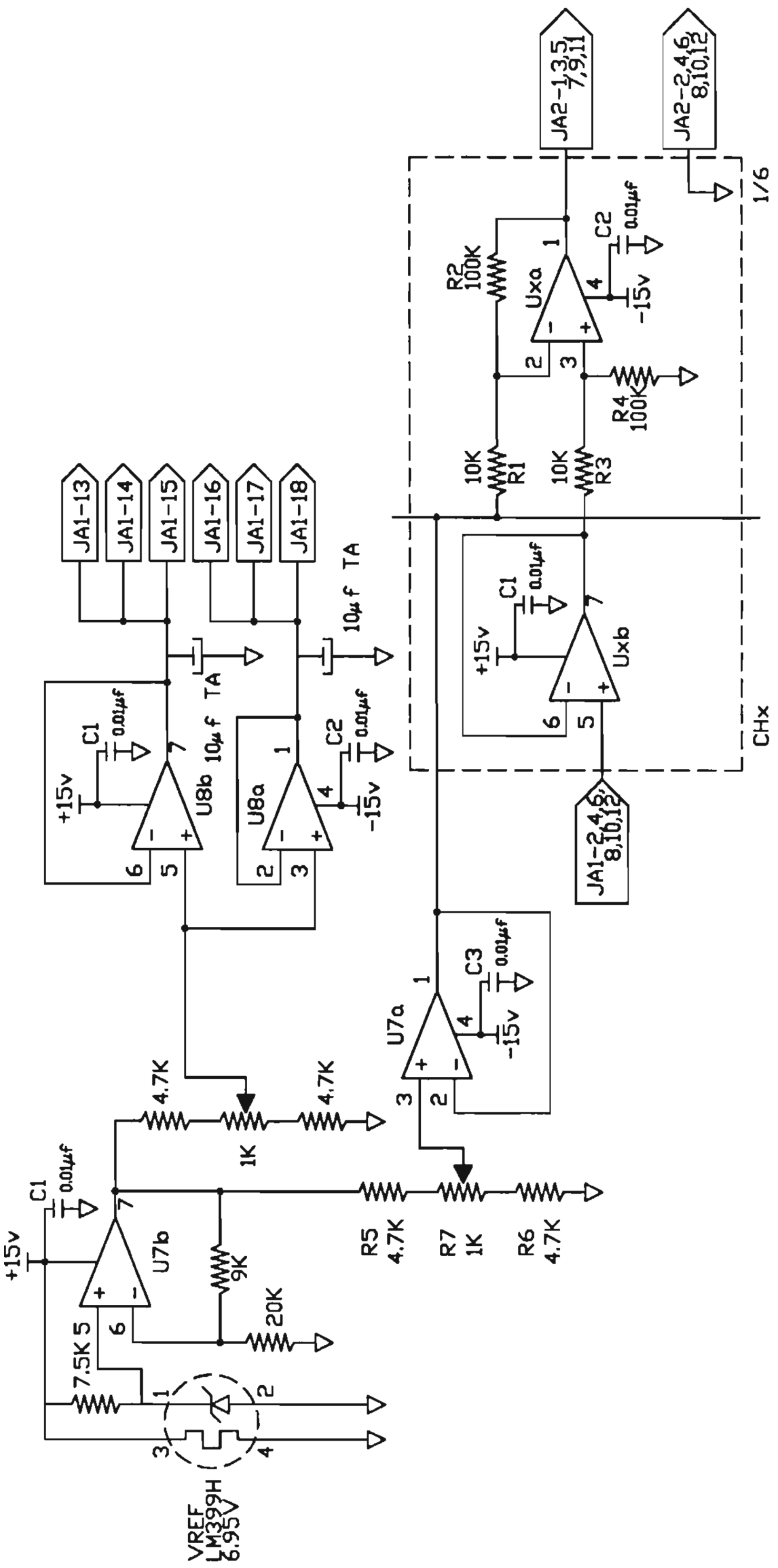


Figure 25: Signal Conditioning Board

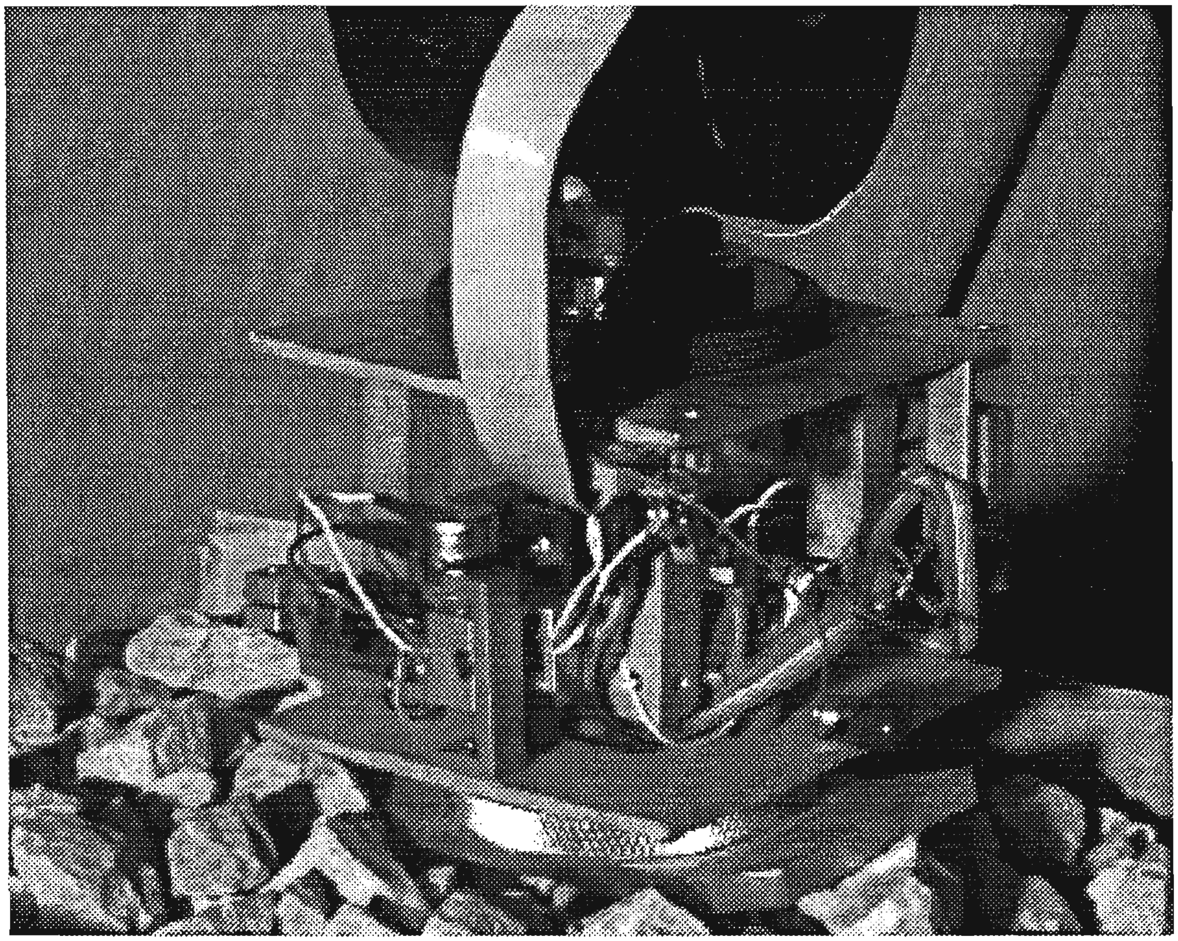


Figure 26: Exploration and locomotion application (courtesy, P. Sinha)

7 Conclusion and Future Work

The wrist outlined here is currently in use in the GRASP lab. Projects using the wrist include "Teleprogramming: Towards Delay-Invariant Remote Manipulation" [2], "Robotic Exploration of Surfaces with a Compliant Wrist Sensor" [6], and "Simlifying Tool Usage in Teleoperative Tasks" [4]. Figure 26 illustrates the previous wrist design in use exploring the environment[4]. Figure 27 shows the robot/wrist holding an impact wrench, undoing a bolt. Figure 28 shows the Penn Hand attached to the wrist, using artificial intelligence to learn object grasping techniques. The wrist has been shown to be a useful force/torque sensor for hybrid position/force control implementations[3].

Two major improvements to the wrist would improve the usefulness and accuracy. First, determination of the complete 6x6 stiffness matrix for the wrist would both improve the accuracy of force control algorithms and characterize the effects of the coupling of orthogonal forces. This would lead to more accurate control. Second, the accuracy of the sensing linkage could be improved by using resolvers instead of potentiometers at the linkage joints, or possibly substituting a parallel sensing linkage structure using LVDTs as position transducers.

[4]Note that application pictures shown in this section are actual implementations of the previous wrist design

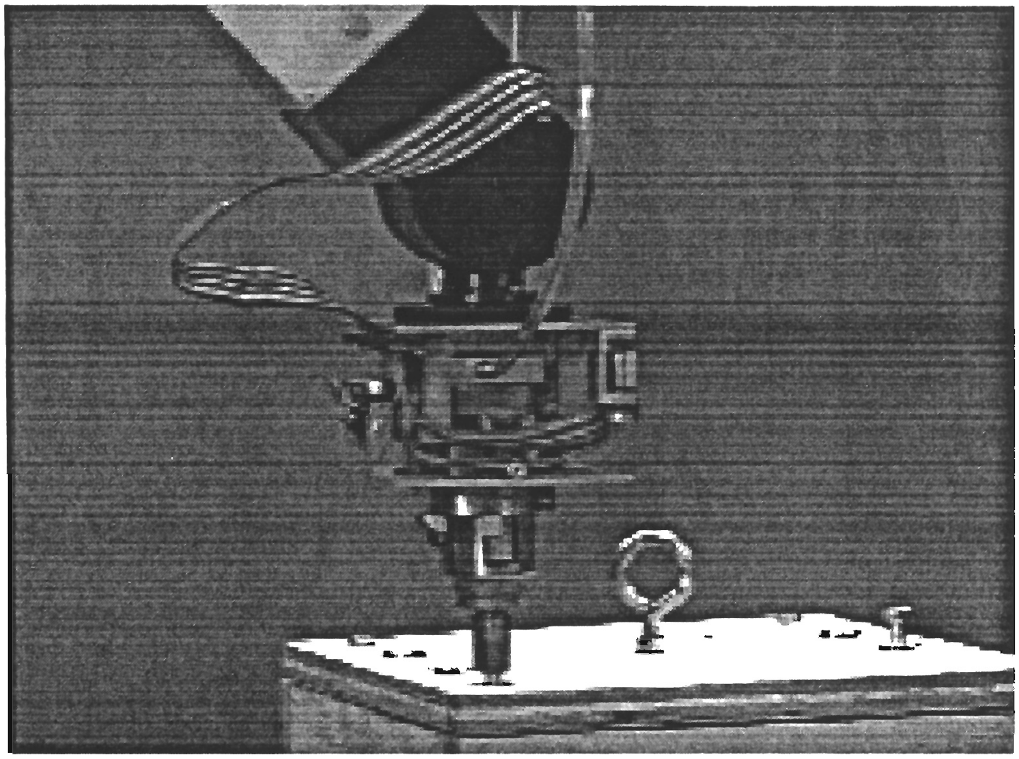


Figure 27: Impact wrench attached to wrist

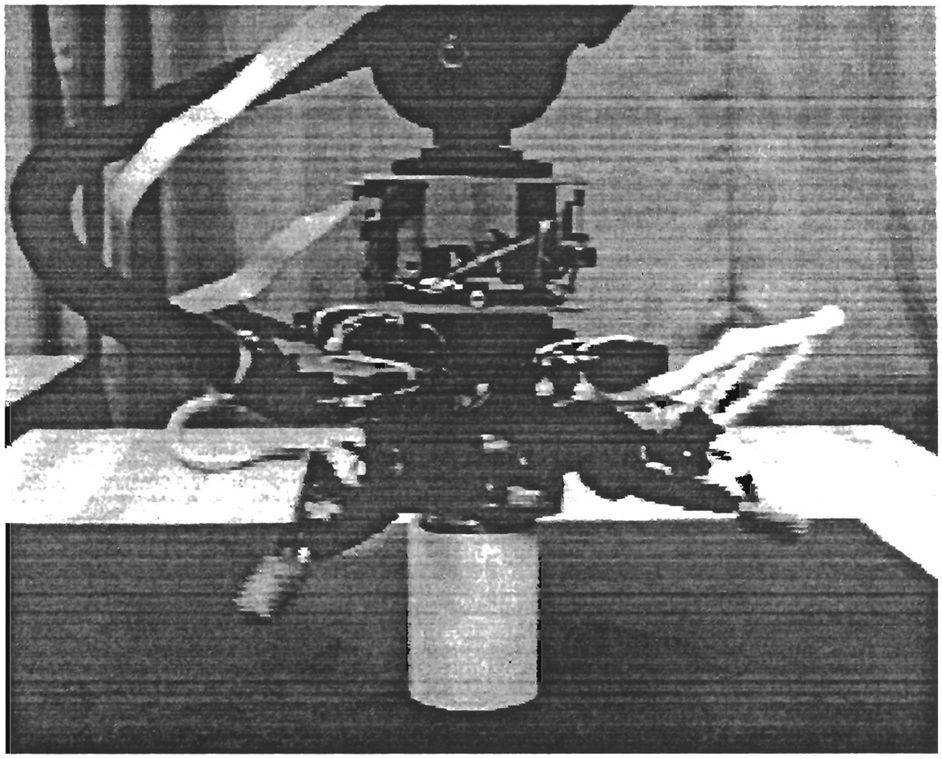


Figure 28: Penn Hand attached to wrist (courtesy, M. Salganicoff)

References

[1] *Vibration and Shock Mount Handbook.* Stock Drive Products, New Hyde Park, NY, 1984. Product Catalog 814.

[2] Janez Funda. *Teleprogramming: Towards Delay-Invariant Remote Manipulation.* PhD thesis, University of Pennsylvania, 1991.

[3] Thomas S. Lindsay. *Teleprogramming: Remote Site Robot Task Execution.* PhD thesis, University of Pennsylvania, 1992.

[4] Thomas S. Lindsay and Richard P. Paul. Simplifying tool usage in teleoperative tasks. In *SPIE Telemanipulator Technology*, November 1992.

[5] Randall K. Roberts, R. P. Paul, and Benjamin M. Hillberry. The effect of wrist force sensor stiffness on the control of robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 269–274, April 1985.

[6] P.R. Sinha, Y. Xu, R.K. Bajcsy, and R.P. Paul. Robotic Exploration of Surfaces using a Compliant Wrist Sensor. *International Journal of Robotics Research*, 12(1), February 1993. To appear.

[7] Richard Volpe and Pradeep Khosla. Experimental verification of a strategy for impact control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1854–1860, 1991.

[8] Yangsheng Xu. *Compliant wrist design and hybrid position/force control of robot manipulators.* PhD thesis, University of Pennsylvania, 1989.

A.3 Operator Interaction and Teleprogramming for Subsea Manipulation

Operator Interaction and Teleprogramming for Subsea Manipulation

Craig Sayers, Richard Paul and Max Mintz

November 1992

This material was presented at the Fourth International Advanced Robotics Program (IARP) Workshop on Underwater Robotics, Genova, Italy, 4-5 November 1992 and is based upon work supported by the National Science Foundation under Grant No. 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.

Abstract

The teleprogramming paradigm has been proposed as a means to efficiently perform teleoperation in the subsea environment via an acoustical link. In such a system the effects of both limited bandwidth channels and delayed communications are overcome by transmitting not Cartesian or joint level information but rather symbolic, error-tolerant, program instructions to the remote site. The operator interacts with a virtual reality of the remote site which provides immediate visual and kinesthetic feedback. The uncertainty in this model can be reduced based on information received from the slave manipulator's tactile contact with the environment. It is suggested that the current state of the model be made available to the operator via a graphical display which shows not only the position of objects at the remote site but also, through the use of color clues, the uncertainty associated with those positions. The provision of uncertainty information is important since it allows the operator to compromise between speed and accuracy. An additional operator aid, which we term synthetic fixturing, is proposed. Synthetic fixtures provide the operator of the teleprogramming system with the teleoperation equivalent of the "snap" commands common in computer aided design programs. By guiding the position and/or orientation of the master manipulator toward specific points, lines or planes the system is able to increase both the speed and precision with which the operator can control the slave arm without requiring sophisticated hardware.

1 Introduction

Teleoperation is the performance of work at a distance [25] and involves an operator controlling the force or displacement of a slave manipulator while receiving both visual and kinesthetic feedback. We wish to be able to perform subsea teleoperation tasks by having a human operator, located either on a boat or ashore, control a manipulator located on an unmanned untethered submersible. Unfortunately, the only suitable long-range underwater communications systems are low-bandwidth, delayed, acoustical links [8]. The problem therefore is to perform a teleoperation task successfully despite the fact that communication delays on the order of seconds exist between the operator and slave sites and bandwidth requirements limit the amount of data which may be transferred between sites to less than 10Kbit/s. The teleprogramming concept, developed by Paul, Funda, and Lindsay, provides a way to overcome these limitations [11, 18, 12].

In such a system it is assumed that sufficient sensor data has been received from the slave site so as to enable the construction of a model of the remote environment. By using this model at the master station we can create a simulation of remote environment with which the operator can interact both visually and kinesthetically. The operator's actions are transformed, in real time, into a sequence of robot program instructions which, when executed by the remote manipulator, seek to mimic the operator's actions.

It is recognized that the initial sensor data will be inaccurate. Small discrepancies between the model and actual world can be accommodated by allowing compliant motions which are within some pre-specified tolerance of the simulated motion. Larger discrepancies can be detected by including force and velocity limits along with the robot program commands for execution by the slave.

In this paper we consider how large discrepancies between the real and simulated worlds might be reduced by making use of information gained from the slave manipulator's kinesthetic interaction with the environment. By making use of a flexible compliant wrist, standard end-effector tools could be used as probes to gather data. However, merely updating the model is not sufficient because the operator also needs some indication of the positional uncertainty of objects. It is proposed that color clues be employed to aid the operator in compromising between the fast, but risky, approach of directly manipulating objects whose position is uncertain and the slow, but safe, method of feeling out the position of each object before attempting to work with it.

We also consider the problem of increasing the precision with which the slave manipulator is operated while simultaneously increasing the speed with which the operator can control it. To solve this apparent contradiction we propose employing "synthetic fixtures" where the system actively guides the motion of the master arm along one, or more, degrees of freedom such that it conforms to pre-defined and task-dependent geometric primitives.

2 Teleprogramming

The problems introduced by a significant communications delay are solved by decoupling the master and slave systems (see Figure 1). The operator interacts, not with the remote manipulator, but instead with a virtual reality of the robot and its task. When the operator moves the master manipulator the graphical image of the slave moves immediately within the virtual environment. Thus the operator has the impression that he or she is controlling a robot without any time delay.

When a collision is detected in the virtual world the operator's commanded motions are filtered so as to prevent further motion in the negative direction of the contact normal. In this way kinesthetic feedback is provided to the operator as the master arm will no longer move to penetrate the surface. The arm may be slid over a contact surface by simply maintaining a force on the master arm in the negative normal direction of that surface. If the arm, or object it is carrying, then comes into contact with other surfaces additional constraints are imposed on operator inputs. Using such a system

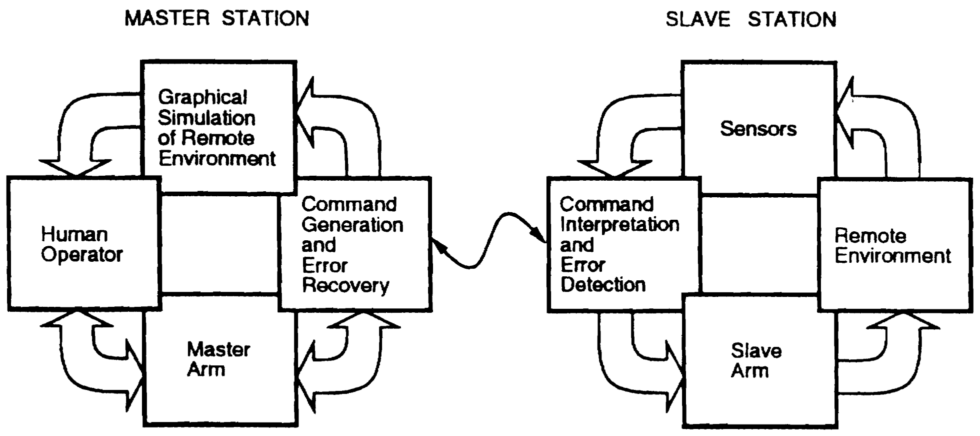

MASTER STATION
SLAVE STATION
Graphical Simulation of Remote Environment
Human Operator
Command Generation and Error Recovery
Master Arm
Sensors
Command Interpretation and Error Detection
Remote Environment
Slave Arm

Figure 1: The teleprogramming system - high bandwidth local feedback loops exist at both the master and slave stations

it is relatively simple to perform tasks such as exploring the inside of a box entirely by feel.

As the operator performs the task at the master station (see Figure 2) his or her task interactions are monitored and translated into robot instructions which are sent to the manipulator at the remote site for execution. These instructions take the form of "execution environments" which specify:

1. A task frame
2. Displacement control and force control
3. Pre-load forces for force control directions
4. Guard forces and velocities
5. The compliance state to assume upon successful completion

Since these commands are at a relatively high level they are well suited to transmission over a low-bandwidth acoustical link. The slave manipulator, by executing these instructions, attempts to mimic the actions of the operator. The instructions which involve contact interaction with the task

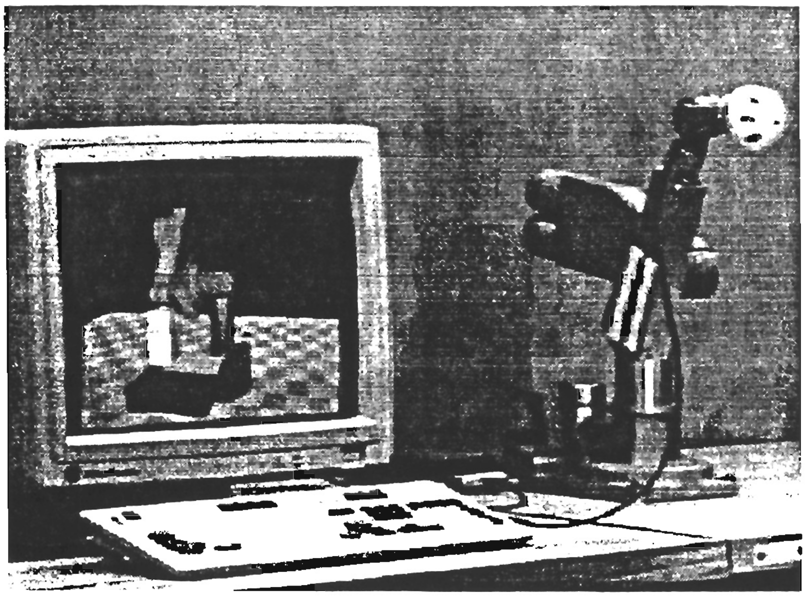


Figure 2: Photo of the master station

are "guarded moves" [17] in which motion or force exertion are terminated by either a reaction force or resulting motion. By continuously comparing its motion with these prescribed tolerances the slave is able to detect cases where its motion significantly differs from that predicted by the master station. When such an error occurs the slave pauses execution and advises the master system which can then reset the graphical view presented to the operator to correspond to the actual position of the slave.

Because operators interact with a virtual reality of the remote world they are, to a large extent, insulated from the effects of the significant communications delay between the master and slave sites. However, when an error is detected at the remote site the slave must transmit a message to the master and then wait until new instructions are received — this will take a minimum of one whole round-trip communications delay. Thus, if we are to perform tasks efficiently it is important that the number of such errors be minimized. This can be achieved using two complimentary approaches. The first is to continuously update the master's model of the slave site as new information becomes available (thus minimizing the discrepancy between the real and virtual worlds). The second method is to make the operator aware of cases where errors are likely to occur by providing a visual indication of uncertainty.

3 Interacting with the World Model

It is generally recognized that the efficient performance of teleoperative tasks in a delayed environment requires the use of some form of predictive or preview display [2, 5, 14]. In the teleprogramming system this is accomplished by maintaining a model of the remote environment at the master station and displaying a graphical representation of that model to the operator.

It is assumed that an initial, imprecise, world model could be constructed prior to the initiation of a teleprogramming session using sensor data collected from the remote site. The uncertainty in this model can then be reduced during the teleprogramming session using information gained from the slave manipulator's kinesthetic interaction with the environment. The emphasis on kinesthetic, rather than other sensors, is motivated by the necessity of operating in real-time over a low bandwidth acoustical link as well as by the relative immunity of physical contact-based sensing to "the vagaries of seawater composition" [13]. Rather than using a specialized touch sensor [21] the intention is that the current end-effector tool be used as a probe [24]. By equipping the slave robot with a flexible compliant wrist [19] it is possible to detect contact between the tool and the environment.

Since data from the remote site will itself be uncertain we can never know exactly the position of any object. Instead the system must combine *a priori* knowledge with available information from the remote world to generate a "best estimate" as to the current state of the environment. Given multiple contacts with an object we can, using a least squares type of approach, generate an updated estimate of its position

For example, consider the simplest case - a single plane. Define the plane by the set of all points, q, which satisfy:

$$\mathbf{q} \cdot \mathbf{v} - \mathrm{k} = 0$$

where $\mathbf{v}$ is a unit normal to the plane and k is the distance from the origin to the closest point on the plane. Now, let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$ be the points at which the plane has been contacted. The distance from each contact point to the plane is then:

$$\mathbf{c}_i \cdot \mathbf{v} - \mathrm{k}$$

Thus an improved estimate of the plane's position may be determined by finding $\mathbf{v}$ and k which minimize:

$$\sum_{i=1}^{N} (\mathbf{c}_i \cdot \mathbf{v} - \mathrm{k})^2$$

subject to the constraint that $\mathbf{v}$ is a unit vector.

This may then be solved as an eigenvalue problem [7, 3].

Since the initial graphical model is imperfect it is not sufficient to merely provide the operator with information about the position of objects in the environment. Consider, for example, the case where an operator is attempting to place a wrench over a bolt head. In such a situation the operator must make a choice between the direct approach of simply applying the wrench to the nut or, alternatively, the more conservative method of feeling around the object first to accurately locate the position of the nut before attempting to place the wrench over it.

To aid the operator in making such a decision we propose using color clues to discriminate between objects with differing levels of uncertainty. These clues could be based directly on the positional uncertainty of each object or alternatively, and perhaps more importantly, they could be based on the probability of successfully interacting with each object. For example, if the operator were to select a torque wrench the system could color bolt heads according to the probability that the wrench could be successfully placed over them.

By continuously updating the world model and by displaying uncertainty information to the operator we can guide them in choosing where next to move the slave robot. However, merely helping the operator choose where to move next is not enough - we must assist the operator in actually moving to those chosen locations. Synthetic fixtures provide just such an operator aid.

4 Synthetic Fixtures

We are interested in designing a system which will allow an operator to control the slave manipulator in a very precise manner (as may be required, for example, when parts are to be mated together). One approach to this problem is to improve the visual information avaliable to the operator. This could be achieved by using stereo systems [9], by providing additional visual clues with the addition of shadows and textures [26] and by superimposing visual enhancements onto the viewed scene [16]. A second, complimentary, approach is to improve the "feel" of the master arm by using improved hardware [20]. The problem is that no matter how realistic the graphical simulation is and no matter how perfect we make the master arm we will still be limited by the accuracy of the human operator.

Consider, by way of example, the situation where an operator wishes to move the slave robot along a straight line in Cartesian space - as may be required during the task of mating two parts. The problem is that its very difficult for the operator to move the master arm in a perfectly straight line - even if the problem is reduced to only two dimensions it is still very hard - consider trying to draw a straight line without using a ruler. One possible solution is to have the operator define starting and ending points and then have the system generate commands for motion between them. This could be accomplished with the "position clutch" described by Conway et al [6]. However the operator must still correctly specify the starting and ending points.

This is, in some sense, analogous to the problem faced by computer aided design (CAD) programs where its very difficult, for example, for a user to draw a line which exactly meets another line. In CAD systems this is overcome by the use of a "pseudo pen location" [23] or "precision point assignment" [15] or, more recently, the "osnap" command [1]. These solutions to the precision problem all work by moving the cursor, not to exactly where the operator pointed, but rather to where the system thinks the operator intended.

What is needed for teleoperation is similar feature which provides kinesthetic as well as visual feedback.

We propose using a concept we term synthetic fixturing to bring these ideas to the telerobotics domain. In essence we suggest giving the operator a kind of virtual ruler - a device which allows the operator to feel, as well as see, the relationship between the current end-effector position and a number of pre-defined task-dependent geometric primitives. Since we are working in a computer-generated virtual environment we're not limited just to physically realisable rulers. For example we should be able to assist the operator in maintaining a spatial position and/or orientation within a particular plane, or along a specified line, or within a particular region. In many cases it is unnecessary for the operator to explicitly request that a synthetic fixture be activated since the system can infer which fixtures are appropriate based on the location and current state of the end effector. For example, when the operator moves a torque wrench near a bolt the system could activate a fixture to guide the wrench along a line normal to, and centered on, the top surface of the bolt.

Tactile feedback has been employed in a number of telerobotic systems, however these have typically focussed on providing the operator with forces derived from the physical interaction between objects. These forces were either attractive, as in the case of molecular docking [22] or repulsive, as in the case of collision avoidance [4]. Synthetic fixturing differs from these in that it presents the operator with task-dependant tactile clues which have no direct physical analogy.

It is important that the addition of the fixturing capability not impose undue restrictions on the operator. It should be possible, for example, for the operator to move a torque wrench past a bolt without being forced into a vertical position over it. Thus, we would like a system which compromises between providing as much aid as possible to the operator when its needed and yet is as unobtrusive as possible when not required.

Fixturing is accomplished by giving the manipulator a tendency to drift, in one or more degrees of freedom, toward a predetermined value. The trick is to make the force sufficiently large that an operator who wishes to make use of the fixture can just relax and let it pull their hand along and yet sufficiently small that an operator who wishes to move in a different direction can still get there - they just need to push a little harder.

In our implementation the master manipulator is position controlled. When fixturing is not active the arm is servoed by reading the force of the operator (using a six-axis wrist-mounted force/torque sensor) and computing a new Cartesian set point for the arm motion based on those readings. Fixturing is implemented by computing the distance from the end-effector to the fixture and then altering the set point based on that distance. For example, consider the case of a planar fixture where orientation is not controlled.

The fixture plane is defined by the set of all points $\mathbf{q}$ which satisfy:

$$(\mathbf{q} - \mathbf{p}) \cdot \mathbf{v} = 0$$

where $\mathbf{p}$ is some point on the plane and $\mathbf{v}$ is a unit normal to the plane. In this case the displacement, $\mathbf{h}$, from the calculated set-point, $\mathbf{e}$, to the plane is just:

$$\mathbf{h} = ((\mathbf{e} - \mathbf{p}) \cdot \mathbf{v})\,\mathbf{v}$$

A modified set point is then calculated (see Figure 3) with the simple function:

$$\mathbf{e}' = \mathbf{e} - \mathbf{h}\frac{\mathbf{a}}{\mathbf{a} + \mathbf{h} \cdot \mathbf{h}}$$

The actual choice of fixturing function is somewhat arbitrary but the above equation provides stable operation with the desirable property that the ap-

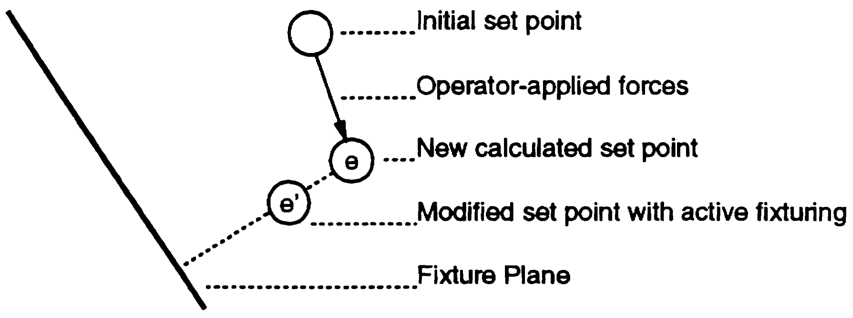


Figure 3: A planar synthetic fixture

parent force felt by the operator increases as the end effector moves closer to the fixturing plane.

It seems intuitively clear that the speed with which an operator can move will increase as the required accuracy decreases and indeed this has been found to be the case. The time taken for an average human to perform a motion is approximately proportional to $\log(d/W)$, where d is the distance to be moved and W is a measure of the desired accuracy [10]. Thus, in the teleprogramming system, we can increase the rate at which an operator can work by decreasing the accuracy with which they must move. The use of synthetic fixtures allows us to do this without compromising the positional accuracy of the slave.

Synthetic fixturing has an additional, more subtle benefit. As the system observes the operator's actions it must interpret the operator's input and transform that into a command sequence for execution at the remote site. Now, if the operator complies with, for example, a line fixture then motion of the master will be along a straight line and its obvious to the system that this motion was what the operator desired. In the case where the operator chooses to deviate from the fixture path this will also be obvious and the system can act accordingly. Without fixturing the system must guess whether any deviation from a straight line is deliberate or merely accidental.

5 Conclusions

The teleprogramming paradigm has been suggested as a means to efficiently perform teleoperative tasks in the uncertain subsea environment. It is assumed that an imprecise model of the remote site is known before the teleprogramming session is initiated. That model can them be improved by using information gained from the remote manipulator's kinesthetic interaction with the environment. By making use of an instrumented compliant wrist it is possible to obtain simple contact information using standard end-effector tools.

The use of color clues to provide the operator with a visual measure of uncertainty has been proposed. Providing users with information about both the position and positional uncertainty of objects should enable them to compromise between the fast, but risky, approach of directly manipulating objects whose position is uncertain and the slow, but safe, method of feeling out the position of each object before attempting to work with it.

Improvements to the master station for teleoperation systems have generally focussed on improving operator performance by providing more sophisticated master arms or improved visual displays. We propose synthetic fixturing, where the master system actively guides the operator's motions in one or more degrees of freedom, as a means of increasing precision and speed without the need for sophisticated and expensive hardware.

The advantages of providing a synthetic fixturing capability for the master station seem quite clear - it aids the operator when needed and yet is relatively unobtrusive when not required. By observing operator motions the system can activate appropriate fixtures automatically and thus there should be little need for additional operator intervention. The advantages of adding color clues are, however, considerably more difficult to quantify and much more testing is required to evaluate the merits of this operator aid.

References

[1] Autodesk Inc. *AutoCAD Reference Manual*, 1989.

[2] Anatal K. Bejczy, Steven Venema, and Won S. Kim. Role of computer graphics in space telerobotics: Preview and predictive displays. In *Cooperative Intelligent Robotics in Space*, pages 365–377. Proceedings of the SPIE vol.1387, November 1990.

[3] Ruud M. Bolle and David B. Cooper. On parallel Bayesian estimation and recognition for large data sets, with application to estimating 3-D complex-object position from range data. In *Computer Vision for Robots*, pages 90–100. SPIE, December 1985.

[4] Guy Bruno and Matthew K. Morgenthaler. Real time proximity cues for teleoperation using model based force reflection. In *Cooperative Intelligent Robotics in Space II*, pages 346–355. SPIE Vol.1612, November 1991.

[5] Forrest T. Buzan and Thomas B. Sheridan. Model-based predictive operator aid for telemanipulators with time delay. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 138–143. IEEE, November 1989.

[6] Lynn Conway, Richard Volz, and Michael Walker. Tele-autonomous systems: Methods and architectures for intermingling autonomous and telerobotic technology. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1121–1130. IEEE, March 1987.

[7] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[8] Robin M. Dunbar et al. Autonomous underwater vehicle communications. In *ROV '90*, pages 270–278. The Marine Technology Society, June 1990.

[9] E. R. Ellis. Nature and origins of virtual environments: A bibliographical essay. *Computing Systems in Enginerring*, vol. 2, no. 4, pages 321–347, 1991.

[10] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, vol. 47, pages 381–391, June 1954.

[11] Janez Funda. *Teleprogramming: Towards delay-invariant remote manipulation*. PhD thesis, University of Pennsylvania, 1991.

[12] Janez Funda, Thomas S. Lindsay, and Richard P. Paul. Teleprogramming: Toward delay-invariant remote manipulation. *Presence*, vol. 1, pages 29–44, Winter 1992.

[13] A. R. Greig and D. R. Broome. Applications of touch sensors in the subsea environment. In *ROV '90*, pages 137–142. The Marine Technology Society, June 1990.

[14] G. Hirzinger, J. Heindl, and K. Landzettel. Predictive and knowledge-based telerobotic control concepts. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1768–1777. IEEE, May 1989.

[15] Timothy E. Johnson. Sketchpad iii - a computer program for drawing in three dimensions. In *Proceedings - Spring Joint Computer Conference (AFIPS)*, pages 347–353, 1963.

[16] Won S. Kim and Lawrence W. Stark. Cooperative control of visual displays for telemanipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1327–1332. IEEE, May 1989.

[17] J. C. Latombe, C. Laugier, J. M. Lefebvre, E. Mazer, and J. F. Miribel. The LM robot programming system. In Hideo Hanafusa and Hirochika Inoue, editors, *Robotics Research: Second International Symposium*, pages 377–391. MIT Press, August 1984.

[18] Thomas S. Lindsay. *Teleprogramming: Remote Site Robot Task Execution.* PhD thesis, University of Pennsylvania, 1992.

[19] Thomas S. Lindsay, Janez Funda, and Richard P. Paul. Contact operations using an instrumented compliant wrist. In *Second International Symposium on Experimental Robotics*, 1991. Toulouse, France.

[20] Douglas A. McCaffee. Diverse applications of advanced man-telerobotic interfaces. In *Technology 2000*, pages 350–360, March 1991.

[21] Howard R. Nicholls and Mark H. Lee. A survey or robot tactile sensing technology. *The International Journal of Robotics Research*, vol. 8, pages 3–30, June 1989.

[22] Ming Ouh-young, David V. Beard, and Frederick P. Brooks, Jr. 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*, pages 1462–1466. IEEE, May 1989.

[23] Ivan E. Sutherland. Sketchpad - a man-machine graphical communication system. In *Proceedings - Spring Joint Computer Conference (AFIPS)*, pages 328–346, 1963.

[24] T. Tran and A. Giraud. An interactive graphical control system for contact tasks. Technical Report LAAS 90199, Laboratoire d'Automatique et d'Analyse des Systems, Toulouse, France, June 1990.

[25] Jean Vertut and Philippe Coiffet. *Robot Technology*, volume 3: Teleoperations and Robotics. Prentice-Hall, 1986.

[26] Leonard R. Wanger, James A. Ferwerda, and Donald P. Greenberg. Perceiving spatial relations in computer-generated images. *IEEE Computer Graphics and Applications*, vol. 12, pages 44–58, May 1992.

Chapter 5

Parsing and Execution of Semi-Autonomous Commands

The input to the remote system from the master consists of a stream of execution environments, each containing the commands necessary for a basic motion or action. The language parser/interpreter converts these commands into the robot control language. Interpreting the generally simple commands from the master becomes a complex task as the remote system must:

- Determine actual force and guard parameters from approximate or relative magnitude information sent by the master
- Determine the criterion for meeting tolerance conditions
- Determine stopping conditions (collision detection) from local environmental conditions
- Reject noisy data caused by varying masses/surface friction/etc.

The command language itself is simple and straightforward. Free space motions can be interpreted directly into the local robot control language. Motions into contact and motions within contact are more complex, as tolerance checks must be performed, and guard and force conditions, which are commanded only as approximate or relative magnitude information, must be converted into actual values. The guard information is handled by the stopping conditions, discussed in section 5.4. Force information is currently interpreted as constant values modified slightly to facilitate motions in contact with the environment.

One of the important specifications imposed upon the remote system, as mentioned in chapter 3, is that the motions must keep pace with the master arm, so there is no increase in the time lag between the master and remote system. In general, however, a guarded move, which implies uncertainty, can not be executed in the same time as the master motion. For example, a motion into contact with a surface may be expected to travel 2 cm in 2 seconds. If, additionally, there is a tolerance of 2 cm (which is currently implemented in the system), it is possible that the manipulator must travel 4 cm to reach the wall. Traveling at the commanded velocity, this motion will be executed in 4 seconds, double the expected time, and the time lag will increase. This problem is discussed in this chapter, and solutions are presented.

Various methods may be used to determine stopping conditions from local environmental conditions. Experimental procedures (EPs) could be used to determine local environmental conditions [Sinha 1991], but this would require extra time, and cause the remote system to lag further and further behind the master. However, the remote site does not function well with simple constant parameters; parameters which vary with current sensory data are required. Experiments have shown that increasingly adverse conditions, such as higher surface friction and larger payload masses tend to increase the level of noise in sensor feedback. Therefore, the way the system compensates for dynamic parameters is to reject the noisy sensor data they present, and to determine sensor events, such as collisions with the environment, by large changes in the fluctuating data. Statistical methods are used for this purpose, with the result of reliable stopping conditions and collision detection.

The interpretation and execution of commands for the teleprogramming remote site system, developed as part of this research, will be outlined in this chapter. The specific problems of tolerance checks and stopping conditions will be presented, including experimental results.

5.1 Background

The concept of the guarded move was evident in Ferrell and Sheridan's supervisory control: "At the primitive level, the language is constructed around a basic action: a movement in a given direction terminated on the achievement of specified sensor states, and/or a given distance moved." [Ferrell and Sheridan 1967], and this is implemented in the MANTRAN language [Barber 1967]. Peter Will coined the phrase *guarded move* as "a move until some expected sensory event occurs" [Will and Grossman 1975]. It is important to note, though, that unexpected sensory states must also be monitored, to prevent potentially damaging events. In both of the cases cited above, the occurrence of a sensory state leads to branch statements. Because of the impossibility of recognizing every possible sensor state and creating a branch for each, and the fact that each branch may also rely on a guarded move, the idea of branching has been eliminated in teleprogramming. If a guarded move succeeds (desired sensor states are reached), a programmed post motion command sequence is executed, and the remote site continues on with the next command. If a guarded move fails (undesired sensor states are reached), the robot stops all motion, sends information about the error state and current manipulator position back to the master, and awaits a corrective command sequence.

The command execution stage also requires some adaptation to the remote environment. As Ferrell and Sheridan state, "Simply because predictions often are not borne out and environments change, such feedback as there is [at the remote site] must be able to modify the internal representation of the environment. In at least this elementary sense, the system must be able to learn from experience" [Ferrell and Sheridan 1967]. This is applicable to the research proposed here, in that the master system does not have any dynamic or real-world information about the remote site. The remote model of surface properties, for example, must be modified as information about the surface is gathered. Thus, the system is able to adapt to changing conditions in the remote site environment.

Task Frame Management

DefineVector(*name*; $< v_x, v_y, v_z >$: *ref-frame*)
DefineTaskFrame(*name* : *ref-frame*; *origin*; *x-axis*; *y-axis*; *z-axis*)
UseFrame(*frame*)

Force Control

AssignMode(X,X,X,X,X,X) where X $\in$ (F,P)
Force($< v_x, v_y, v_z >; < \tau_x, \tau_y, \tau_z >$)
GuardForce($< v_x, v_y, v_z >; < \tau_x, \tau_y, \tau_z >$)
GuardVelocity($< v_x, v_y, v_z >; < \omega_x, \omega_y, \omega_z >$)

Motion Commands

Move(t ; $< p_x, p_y, p_z >; < \phi_x, \phi_y, \phi_z >$)
Slide(t ; $< p_x, p_y, p_z >$)
Pivot(t ; $< \phi_x, \phi_y, \phi_z >$)

Special Commands

UseTool(*toolname*)
Grasp()
Release()

Table 5.1: The Teleprogramming Command Language

5.2 Command Language

Programs sent to the remote site are made up of execution environments. Each execution environment contains information about the task frame, hybrid modes, force preloads, guards, motions, and post-motion default parameters. Each environment may be simplified by using default parameters from previous environments if no changes are necessary. Generally, the most complex environments are needed only when contact states change.

5.2.1 Command Types

The current language command set is presented in table 5.1. The commands are sorted into four categories. The task frame management commands allow arbitrary frames to be defined for use as the hybrid control task frame. The force control commands are used for assigning the force/position mode directions for the hybrid control, to assign forces that the manipulator is to exert, and to set the guards for guarded moves. These guards can be either force guards, where the manipulator moves until it senses a given force, or velocity guards, where the manipulator stops if velocity limits are violated. In addition to these guards, there exist maximum force and velocity limits that the manipulator cannot violate.

Motion commands are used to move the manipulator. When used in combination with a force or velocity guard, the motion commands also imply a tolerance limit. If the guard is violated before the tolerance is reached, an error message is generated. Similarly, if a tolerance is passed without violation of the guard limits, an error message is generated.

Finally, the special commands are used to control tools. The UseTool() command identifies the tool being used, which may change the function of other commands (see chapter 6). Grasp() and Release() are used to control a gripper.

5.2.2 Sample Execution Environment

An example of a typical execution environment is shown below. As noted previously, many execution environments are much simpler, because they use default task frames, force preloads, etc. from previous environments.

```
0.0   >> Execution Environment #0
0.1   DefineVector(CP;<0.0,0.0,220.0>:EE)
0.2   DefineVector(X;<1.0,0.0,0.0>:EE)
0.3   DefineVector(Y;<0.0,1.0,0.0>:EE)
0.4   DefineTaskFrame(N:EE;CP;X;Y;?)
0.5   UseFrame(N)
0.6   AssignMode(P,P,P,P,P,P)
0.7   GuardForce(<0.0,0.0,-1.0>;<0.0,0.0,0.0>)
0.8   Move(5.0;<0.0,0.0,4.0>;<0.0,0.0,0.0>)
0.9   AssignMode(P,P,F,P,P,P)
0.10  Force(<0.0,0.0,1.0>;<0.0,0.0,0.0>)
```

The first five commands of the execution environment define a new task frame for use. Command 0.1 is used to define an origin for the frame, and commands 0.2 and 0.3 define two of the three axes for the frame. Notice that each of these three vectors is with respect to the frame EE, which is a pre-defined frame located at the end of the robot manipulator (End Effector frame). Once a frame has been defined, other vectors and frames can be defined with respect to it. The task frame itself is created with command 0.4. The newly created frame is then designated for use in command 0.5. Command 0.6 sets all hybrid control modes to position control, which is common for free space motion. A guarded move is defined in commands 0.7 and 0.8. The manipulator will move on the z-direction of the task frame, and expect to stop on a force within a predefined tolerance of the specified motion. If and when this force is sensed, commands 0.8 and 0.9, the post-motion commands, will come into effect. The hybrid modes are reset to reflect the current contact state, and a force is exerted in the normal direction to the surface. If the expected force is not encountered, these post-motion commands are ignored and the remote system sends an error message to the master and waits for new instructions.

If this execution environment is successful, the following execution environment can be carried out, which may make use of the current task frame and the post motion hybrid modes and forces as defaults.

5.2.3 Real World Parameters

A close examination of the commands presented in section 5.2.1 reveals that there is no reference to masses, friction, or any other real-world parameters. The master operator works in a purely kinematic model of the world, and has no knowledge of information such as surface roughness at the remote site. The remote system, however, must be able to interact with the remote environment and thus must be able to compensate for these parameters.

Commands such as Force() and GuardForce() imply real-world parameters as they are parsed. The GuardForce() command is very sensitive to contact states and surface conditions. As the manipulator slides along a surface, more information about the surface can be gathered, and GuardForce() can use this information. This is possibly the most important aspect of the command execution level: the ability of the system to learn from the tasks currently executed in order to respond more intelligently to the sensor inputs. Section 6 will discuss how models of the current sensor inputs are used in order to identify abrupt changes (such as an impact with the surface) from sensor noise. Sensor noise is defined roughly as the sum of electrical and mechanical noise, which can include the effects of surface friction and manipulator dynamics.

The Force() command supplies the remote site with a relative magnitude of force to exert. The remote system interprets this magnitude as a constant value modified by the contact state in order to facilitate sliding within contact. With only one contact, the force value is the constant multiplied by the magnitude. With two contacts, each normal force is reduced slightly to reduce the overall force exerted by the manipulator, in order to make sliding with two contacts simpler. The force reduction factor is determined empirically; the current value used is 85%.

5.2.4 Example Program

Below is a simple program generated for execution at the remote site, followed by an explanation of the program commands. The program is designed to move the manipulator, which has been fitted with a cubic end effector, into contact with a surface, slide along this surface, and finally slide into contact with a wall. The execution of this program is illustrated in figure 5.9. It has been modified slightly from the working program for clarity.

```
0.0   >> Execution Environment #0
0.1   UseFrame(EE)
0.2   AssignMode(P,P,P,P,P,P)
0.3   GuardForce(<0.0,0.0,-1.0>;<0.0,0.0,0.0>)
0.4   Move(5.0;<0.0,0.0,19.5>;<0.0,0.0,0.0>)
```

```
0.5   AssignMode(P,P,F,F,F,P)
0.6   Force(<0.0,0.0,1.0>;<0.0,0.0,0.0>)

1.0   >> Execution Environment #1
1.1   Slide(2.0;<0.0,5.0,0.0>)

2.0   >> Execution Environment #2
2.1   GuardForce(<0.0,1.0,0.0>;<0.0,0.0,0.0>)
2.2   Slide(5.0; <0.0,-11.5,0.0>)
2.3   AssignMode(P,F,F,F,F,F)
2.4   Force(<0.0,-1.0,1.0>;<0.0,0.0,0.0>)
```

Execution environment #0 contains the command to move the robot manipulator, which starts out above a surface, into contact with the surface. It also specifies what the manipulator should do at the successful conclusion of the motion. The first step of the program (0.1) is to specify the task frame in which the manipulator is to be controlled. In this case, a pre-defined frame EE is used, which is the frame of the robot end effector, centered at the wrist point of the robot and aligned with the box end effector. Task frames can be constructed using the programming language, but this has not been illustrated here. Hybrid control modes are set with the AssignMode() command (0.2), which specifies that all cartesian directions in the frame EE are to be position controlled. This is the usual mode selection when the manipulator is in free space. The GuardForce() command (0.3) identifies that the manipulator will be expecting to stop on a force in the negative z-direction (in frame EE). The implementation of this command is explained further in section 5.4. Move (0.4) specifies the time for motion (5.0 seconds) and the distance to move the manipulator (19.5 cm.). A tolerance is built into this command, and this will be explained further in section 5.3. Finally, another AssignMode() command (0.5) and a Force() command (0.6) specify how the manipulator is to react after contact has been achieved. In this case, the control modes are switched so that the surface normal direction is force controlled, with a specified loading force, and rotations about the x- and y-axis are force controlled, with a specified torque of zero. With these modes, the box will actually conform to the normal of the surface, even if the box end effector and the surface were not initially aligned properly.

Execution environment #1 commands the robot to slide away from the wall. The command is deceptively simple because the default states from the previous execution environment hold. The task frame is still EE, the hybrid control modes carry over from command 0.5, and the normal force remains applied. The Slide() command is a simplified form of Move(), in that rotations are not specified. The similar command Pivot() can be used when there are no translation elements in a move. The manipulator simply slides in the positive y-direction (away from the wall) 5.0 cm. in 2.0 seconds. No stopping forces are expected.

Execution environment #2 is very similar to execution environment #0. As in execution environment #1, the default task frame, hybrid modes, and forces are still in effect. The manipulator is warned to expect a force in the y-direction, and then commanded to slide towards the wall. After the wall is reached, the modes are changed appropriate to the new contact state, and forces are specified in both surface normal directions.

This simple program will be used to illustrate features of the remote robot system. Now that the use of the teleprogramming language has been described, the implementation of motion commands and guards can be discussed further.

5.3 Tolerance

The human operator works in a model of the remote world, but the model is only accurate to some tolerance, ϵ. Because the model may be built from sonar or video scans of the remote environment, the tolerance may be quite large.

Motion commands Move(), Slide(), and Pivot() must all be able to compensate for tolerance errors. As noted before, the tolerance level is crucial for the effectiveness of the error message generation. However, there is a correlation between the effectiveness of tolerance limits and the generated motion distance. If the expected limit of the

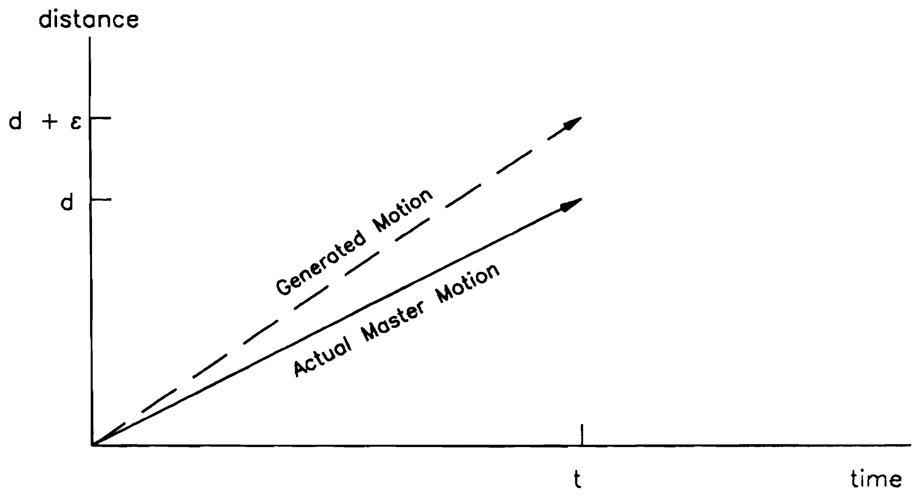


Figure 5.1: Generated Move for Guarded Motion

motion is the same magnitude or smaller than the tolerance, problems will result.

5.3.1 Interpreting the Motion Commands

There are two categories of Move()/Slide() commands. The unguarded move/slide is used for all motions that are not expected to terminate by contacting a surface, such as Execution Environment #1 from the above example. These motions include free space motion and also motion in contact with a surface (general sliding motions). The interpreting of these commands is done simply by computing the cartesian velocity components ($v_x, v_y, v_z, v_{roll}, v_{pitch}$, and v_{yaw}) in units of (mm/interrupt), and converting the time into number of interrupts.

A motion that will be terminated by contact with a surface is called a guarded motion. The motion is guarded in that a relatively small force is expected along the normal to the surface, and motion will stop when this small force is encountered.

It is assumed that the distance to the surface (a wall, for example) is known to a tolerance ($\pm\epsilon$). The master site generates a guarded motion command that executes in time t and moves the manipulator a distance of $d + \epsilon$ in the guarded direction (See figure 5.1).

When the remote site interprets the command

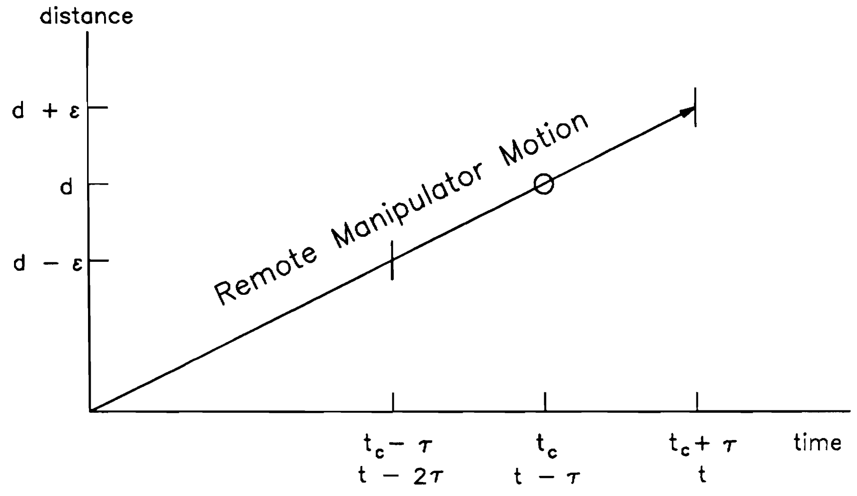


Figure 5.2: Executed Move for Guarded Motion

Move(t_c, d_c)

it first checks if there is a Guardforce() command in the same command packet. If it is determined that the move is indeed a guarded move, the values of d, v, τ, and t are determined as:

$$d = d_c - \epsilon \tag{5.1}$$

$$v = d/t_c \tag{5.2}$$

$$\tau = \epsilon/v \tag{5.3}$$

$$t = t_c + \tau \tag{5.4}$$

The remote manipulator is then commanded to move at velocity v for time t. If motion is terminated by sensing the guarded force after $(t - 2\tau)$ and before t, the motion is determined to be successful (see Figure 5.2).

If t_c is the time of execution of a command for the master, and t_ϵ is the actual time of execution for the remote manipulator,

if $t_c \geq \tau$,

$$t_c - \tau \leq t_e \leq t_c + \tau \tag{5.5}$$

otherwise,

$$0 \leq t_e \leq t_c + \tau \tag{5.6}$$

In terms of d (the expected distance) and ϵ (the pre-defined tolerance) instead of τ,

if $d \geq \epsilon$,

$$t_c(\frac{d-\epsilon}{d}) \leq t_e \leq t_c(\frac{d+\epsilon}{d}) \tag{5.7}$$

otherwise,

$$0 \leq t_e \leq t_c(\frac{d+\epsilon}{d}) \tag{5.8}$$

Examination of $(\frac{d\pm\epsilon}{d})$ shows:

- if $d \gg \epsilon$, the time difference is negligible: $t_e \approx t_c$
- if $d \approx \epsilon$, the time difference may be noticeable: $0 \leq t_e \leq 2t_c$
- if $d \ll \epsilon$, the time difference becomes impossible to work with: $0 \leq t_e \leq \infty$

When t_e is greater than t_c, the delay lag between the master and remote site increases. In order to eliminate the possibility of an unbounded lag, corrective measures must be taken.

When $d \gg \epsilon$, the change in lag time is not a problem. If $d \approx \epsilon$, however, the lag increase can be as much as t_c. This increase in lag can be partially corrected in each of the next moves by decreasing the time proportionally:

$$t'_{i+k} = (K)t_{i+k} \tag{5.9}$$

where $0.0 \leq K \leq 1.0$ (0.9 has been used in experiments), and $k = 1, \ldots, n$. When

$$\sum_{j=1}^{n}(t_{i+j} - t'_{i+j}) = t_{e_i} - t_{c_i} \tag{5.10}$$

the time correction is no longer necessary. If, however, the lag again increases in some of the next k moves, lag time can still increase. If the total time lag becomes greater than T, where T is the communication delay time, a signal should be sent to the master system for it to wait for the remote site to catch up.

When $d \ll \epsilon$, the lag time can increase without bound. In this case, it is appropriate to stop the movement after 2t, and send an error message. The operator can then take appropriate action. However, in order to prevent this situation from arising, post-processing of commands at the master site may be beneficial, in order to combine commands that are generated when a normal-length move is followed by a very short-length guarded move.

5.4 Collision Detection

As the remote manipulator moves around its environment, its most important task is to detect collisions. A collision may signal the correct termination of a command, or may indicate an unwanted interaction with the environment that may damage the manipulator. In the first case, it is important that all mode changes and contact forces are quickly implemented, and the next command is started immediately. In the second case, motion should be immediately terminated, and an error message sent to the master. In both cases, it is important that the collision is detected quickly.

Because the sensor readings have variations due to electrical noise and mechanical vibrations, detecting a sensor event from the standard fluctuations of signals is difficult. When the manipulator is in free space, the sensor noise is relatively small compared with the sensor readings for a collision. When the manipulator is sliding in contact with a surface, however, the noise associated with the sliding friction is quite large. Using statistical methods, the same algorithm can be used to detect collisions in both cases.

A simple statistical algorithm has been developed, based in part upon statistical methods developed for quality control of ongoing processes [Feigenbaum 1983]. This type of algorithm, which is used in manufacturing to determine if a production run

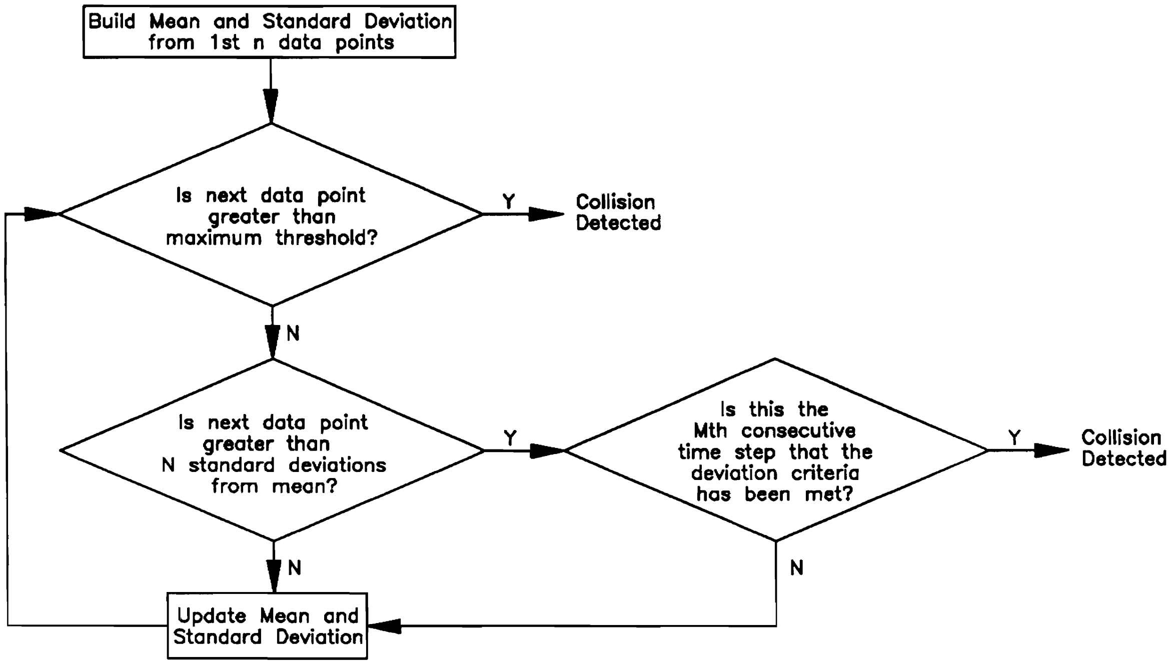

Build Mean and Standard Deviation from 1st n data points
Is next data point greater than maximum threshold?
Y
Collision Detected
N
Is next data point greater than N standard deviations from mean?
Y
Is this the Mth consecutive time step that the deviation criteria has been met?
Y
Collision Detected
N
N
Update Mean and Standard Deviation

Figure 5.3: Force Detection Algorithm

is "out of control" (in the sense that the manufactured parts no longer conform to process or design specifications), assumes that the desired value of the tested parameter remains constant. The methods used in this research relax this assumption slightly, and assume only that the process (sensor readings) mean[1] will attempt to remain constant within a short time scale, or data window. As will be seen in the experimental data, the short history of data does not conform to a normal distribution about a constant mean as well as most statistical textbook examples.

The collision detection algorithm is shown in figure 5.3. The mean and standard deviation are built using a small history of data. The maximum threshold is needed for collisions that fail to be detected by the standard algorithm.

The mean and standard deviation models for the i^{th} data point are built as:

$$\overline{f}_i = \frac{1}{n} \sum_{j=i-n-1}^{i-1} f_j \tag{5.11}$$

[1]The actual value of the mean here is unimportant; the attempt is only to discern changes in sensor readings that lie outside the normal bounds of fluctuations.

$$\sigma_i = \sqrt{\frac{n \sum_{j=i-n-1}^{i-1} f_j^2 - \left(\sum_{j=i-n-1}^{i-1} f_j \right)^2}{n\,(n-1)}} \tag{5.12}$$

where n is the window size (30 data points in current experimentation). When the equation

$$\left| f_i - \overline{f}_i \right| > N\sigma_i \tag{5.13}$$

has been satisfied M consecutive times, a collision is detected. Although this appears to be a very stringent collision detection algorithm, in experiments it is very good at rejecting spurious data and detecting collisions with only a small delay. If there is a normal distribution of the data points about the mean, 99.73% of the data should fall within 3 standard deviations of the mean, until an event occurs which causes larger deviations. However, the distribution of data points in experimental sensor data is quite distorted. The standard deviation is still a useful measure of the general characteristics of sensor data, and suitable values of N and M can be found experimentally.

5.5 Experimental Results

The experimental results presented here pertain mainly to collision detection. First, the natural distribution of sensor data under various conditions is examined; justification of the collision algorithm parameters is based on this data. Next, data collected from execution of the sample program from section 5.5.2 is presented.

5.5.1 Data Distribution

Figure 5.4 shows sensor data collected when the manipulator is stationary. The fluctuations in sensor data here are caused solely by electrical noise. The actual data is shown in figure 5.4(a), and the distribution of this data, in histogram format, is shown in 5.4(b).

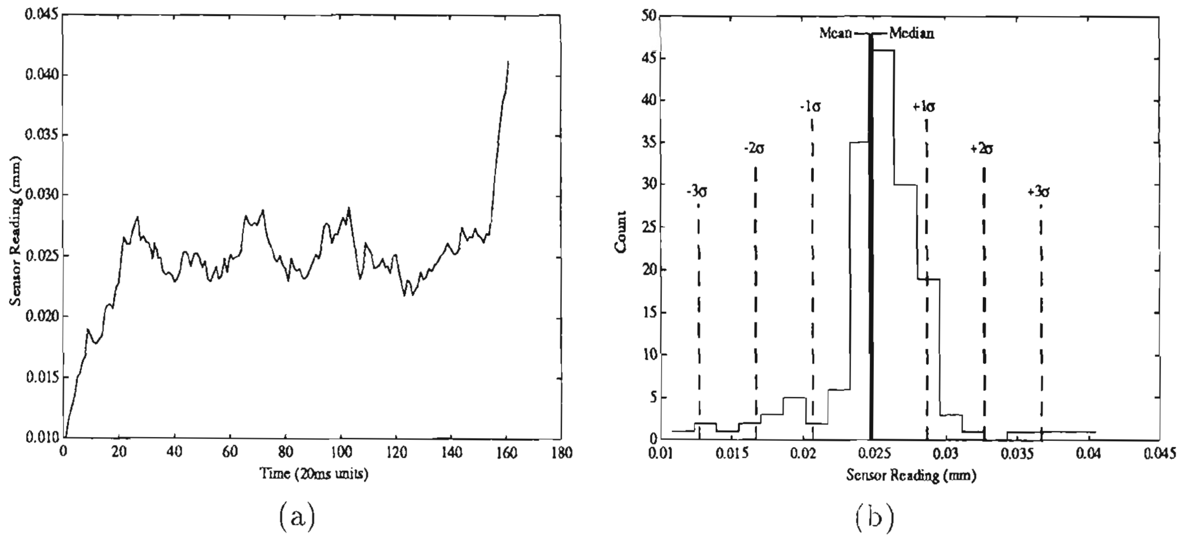


Figure 5.4: Sensor Data from Stationary Manipulator

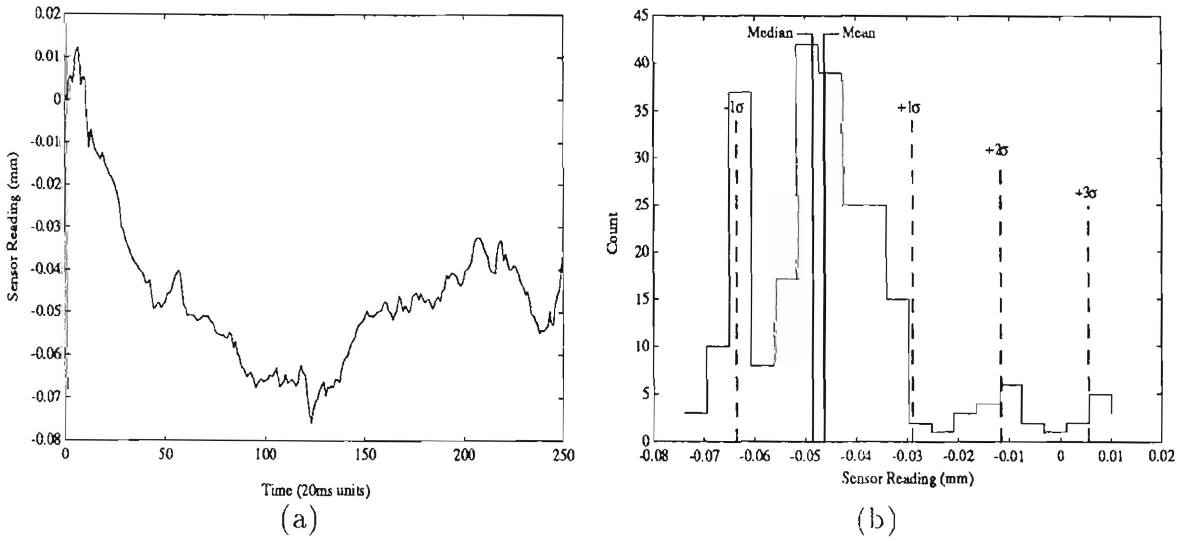


Figure 5.5: Sensor Data from Free Space Motion

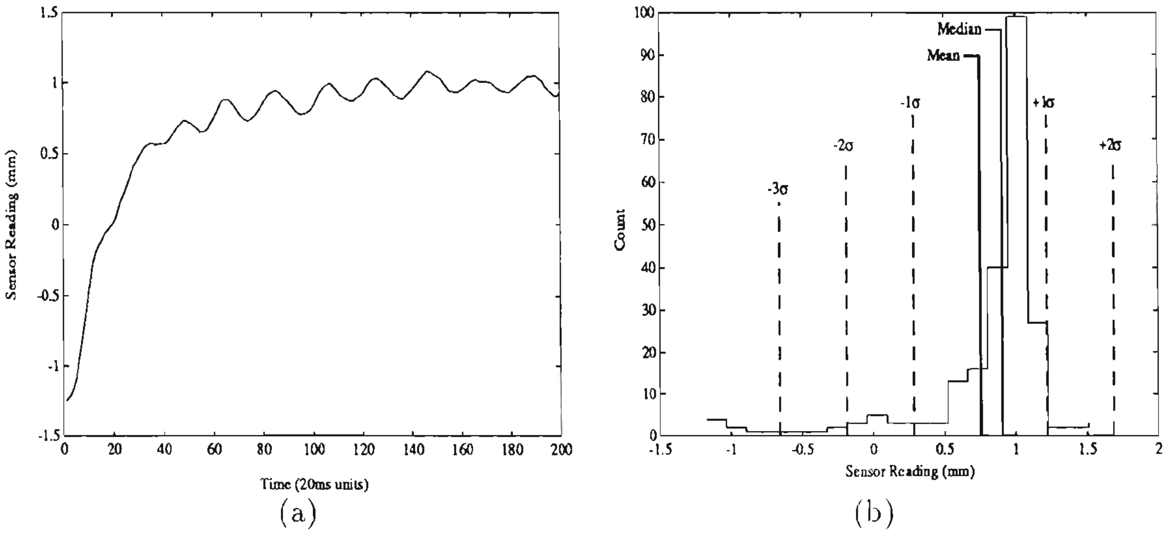


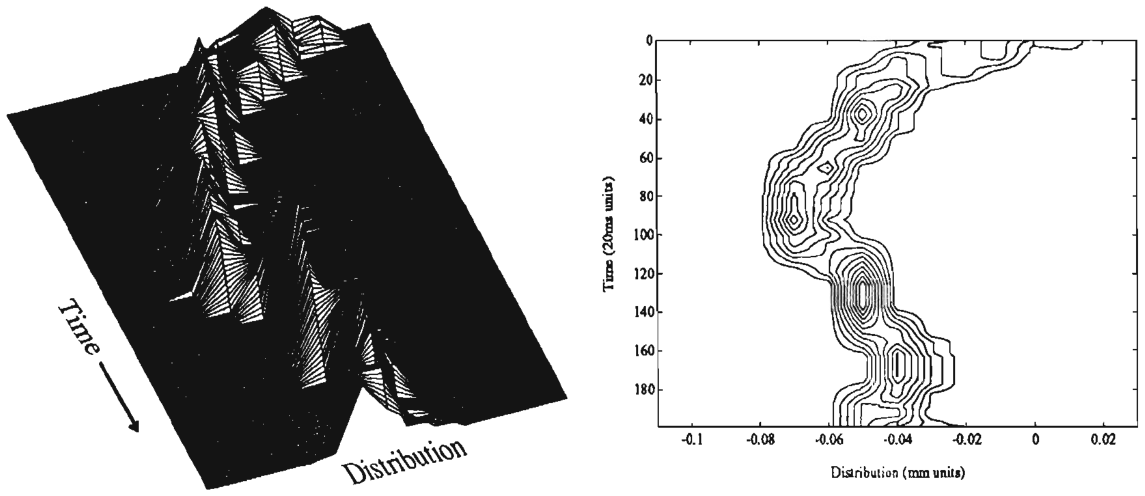
Figure 5.6: Sensor Data from Sliding Motion

Figure 5.5 shows sensor data from a free-space constant velocity motion, with data collected in the direction of motion. The data fluctuations here are caused by both the electrical noise, as above, and the vibration of the wrist/robot system.

Figure 5.6 shows sensor data from a sliding motion, while the manipulator is in contact with a surface. Data again is collected in the direction of motion. The fluctuations present in this data result from the electrical and mechanical noise, as above, as well as additional noise associated with the sliding motion. This sliding noise results from:

- Non-ideal control laws which cause the normal force with the surface to fluctuate slightly.
- Non-homogeneous surface friction.
- Coupling of orthogonal forces.

Tabulated below are the statistical parameters from the three motions described above.



(a) (b)

Figure 5.7: Free Space Sensor Data

	mean	median	σ
no motion	0.0247	0.0249	0.0040
free space	-0.0463	-0.0485	0.0173
slide	0.7510	0.9027	0.4685

Figure 5.7 shows a time history of the distribution of sensor data during a free space motion. Figure 5.7(a) shows the data as a mesh plot, and figure 5.7(b) shows the data as a contour plot. This data illustrates that the distribution varies greatly with time, and is often extremely distorted.

Figure 5.8 shows a time history of the distribution of sensor data during a sliding motion. Although distorted at the beginning of the motion, the distribution during the bulk of the motion is more constant with time, and resembles a normal distribution more than the free space motion.

Based on the experimental data collected and presented above, appropriate values of M and N for the collision detection algorithm have been determined. There is a tradeoff between the reliability of the detection and the force that is exerted before a collision is sensed. Currently, $M = 4$ is used for all collisions. This means that .08 seconds will generally pass between the time of collision and the time of detection.

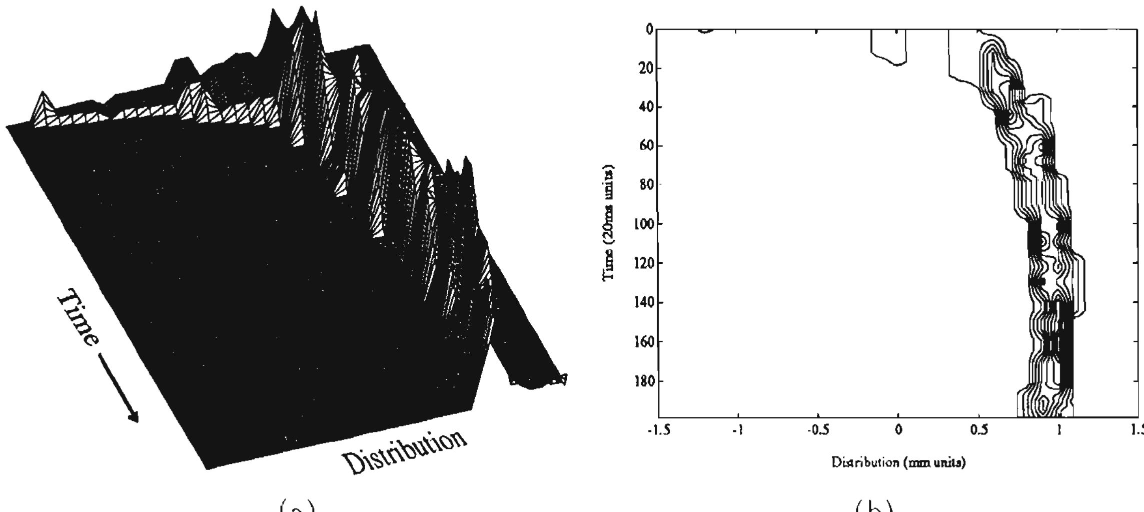


Figure 5.8: One Contact Sensor Data

At high velocities, however, the maximum threshold force may be reached before this time, and motion will stop. For free space into one contact (0 -> 1) collisions, a value of $N = 3.0$ is used. For one contact into two contact (1 -> 2) (and two contact into three contact (2 -> 3) collisions, data not presented here), a value of $N = 2.1$ is used. With these parameters, the reliability is above 90% for 0 -> 1 collisions, and about 80% for other collisions, based upon 50 collision trials for each collision type. When the algorithm fails, the manipulator will stop before the collision occurs. If it violates the tolerance limits, an error is detected and can be corrected; otherwise the manipulator is within a small distance of the surface, and usually can continue with the program without problems. However, this problem needs further investigation.

5.5.2 Example Program

The example program presented in section 5.2.4 is illustrative of tolerance checks and the collision detection algorithm (GuardForce() command). The experiment is shown in figure 5.9. The manipulator, with a cubic end-effector, interacts with a flat surface and a wall.

First, the manipulator executes a guarded move from free space into contact with

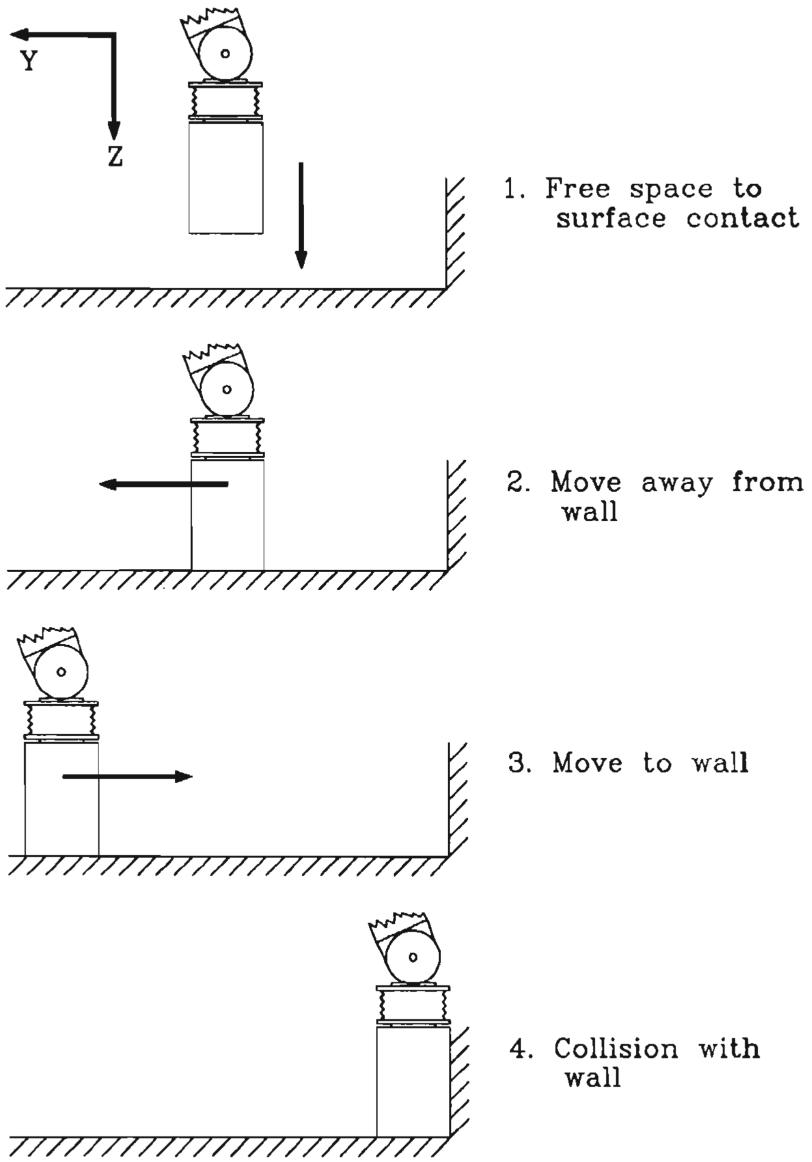


Figure 5.9: Illustrative Experiment

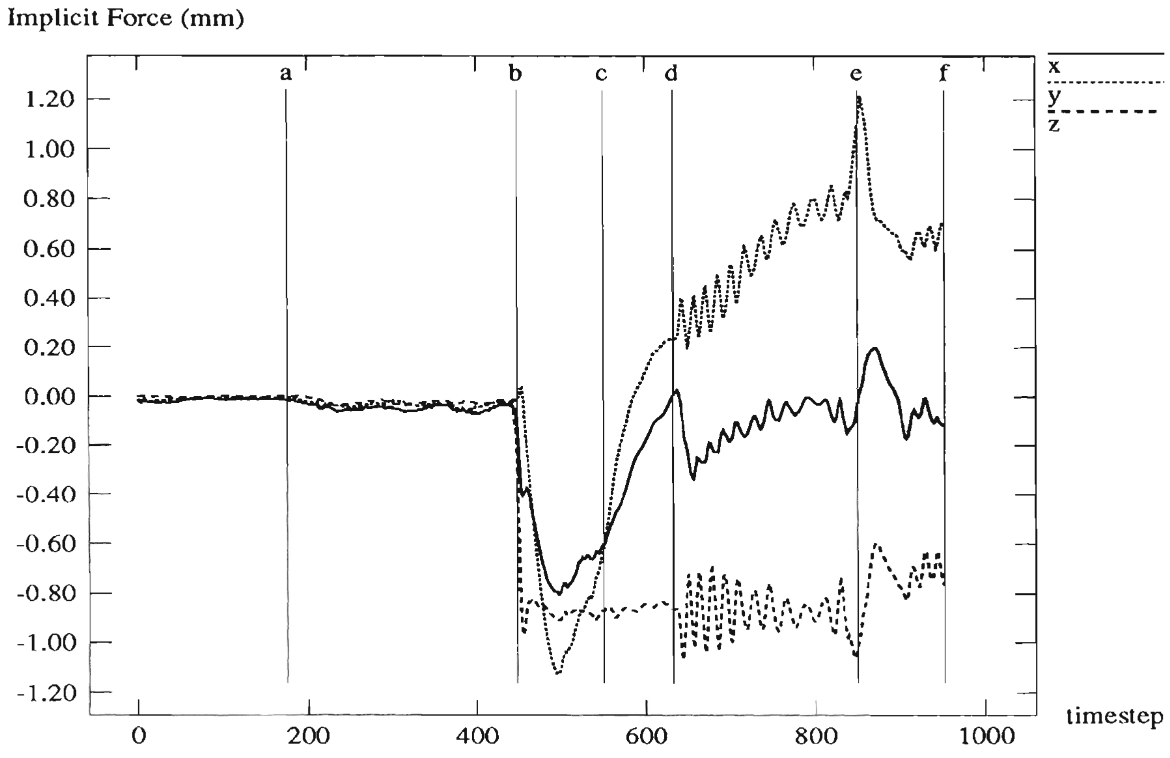


Figure 5.10: Force Data

the surface. Then, the manipulator slides along the surface away from the wall. Finally, another guarded move is executed to bring the manipulator into contact with the wall. This experiment is designed to show the effectiveness of hybrid mode switching (as contact states change), and the collision detection algorithm for a position controlled direction (force detection). Experimental results from the velocity detection algorithm are presented in chapter 6 in reference to tool usage.

Figure 5.10 shows the implicit force data for the entire experiment. Data from (a) to (b) corresponds to the free space motion into contact with the surface, Execution Environment #0 in the sample program and (1) in figure 5.9 (data before point (a) corresponds to no motion of the manipulator). Motion from (b) to (c) corresponds to the move away from the wall, Execution Environment #1 and (2) in figure 5.9. In this section (and all subsequent sections) of motion, the manipulator attempts to maintain a constant contact force in the z-direction. There is no applied force in the x-direction, forces in this direction arise from internal forces in the wrist caused by

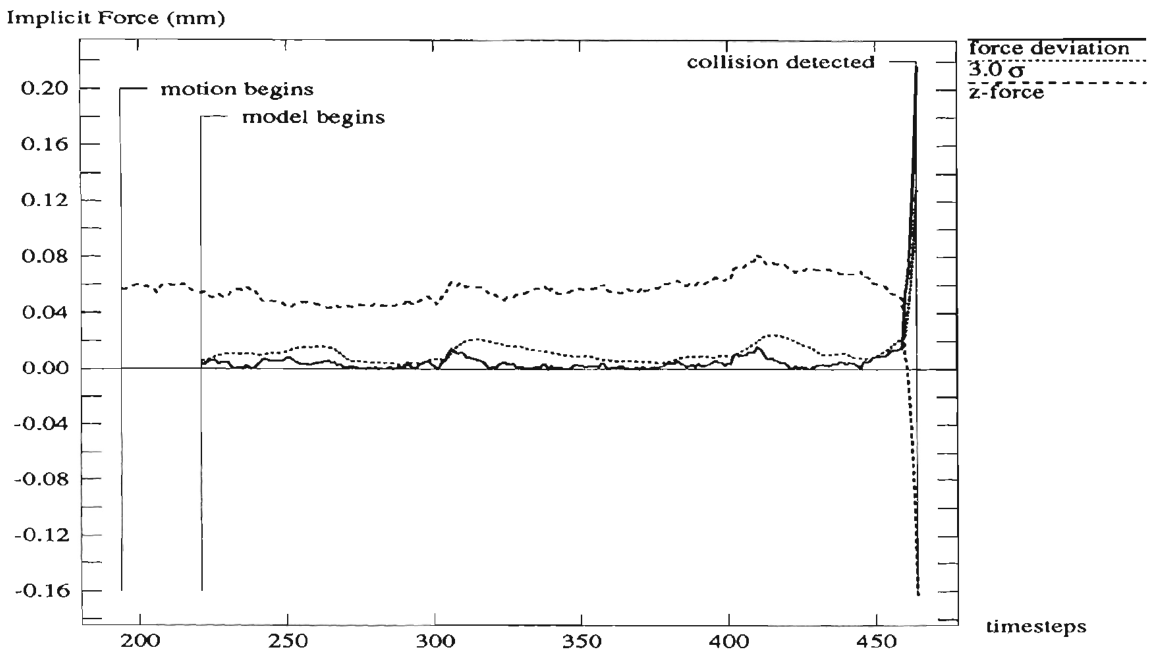


Figure 5.11: Free Space - One Contact Collision

coupling of cartesian directions. Ideally, the force in this direction would always be zero in this experiment. The y-direction force is caused by sliding friction; the force is in the opposite direction to the motion.

Motion from (c) to (d) is a pause, not shown in the example program (implemented as a Move() command with zero motion). Notice that the manipulator forces settle slightly, but do not return to zero values. This illustrates one of the drawbacks of the compliant wrist, in that internal mechanical coupling of the cartesian directions leads to internal forces under common circumstances.

Motion from (d) to (e) is the sliding motion into contact with the wall, Execution Environment #2 in the sample program and (3) and (4) in figure 5.9. Notice again the sliding friction force, opposite in direction to the motion. Finally, motion from (e) to (f) is another pause, while the manipulator is in contact with both the surface and the wall. Applied forces in the z- and y-directions are interpreted as slightly smaller than the z-direction force maintained previously, in order to facilitate sliding motion while in this 2-contact state. The x-direction force is again not intended.

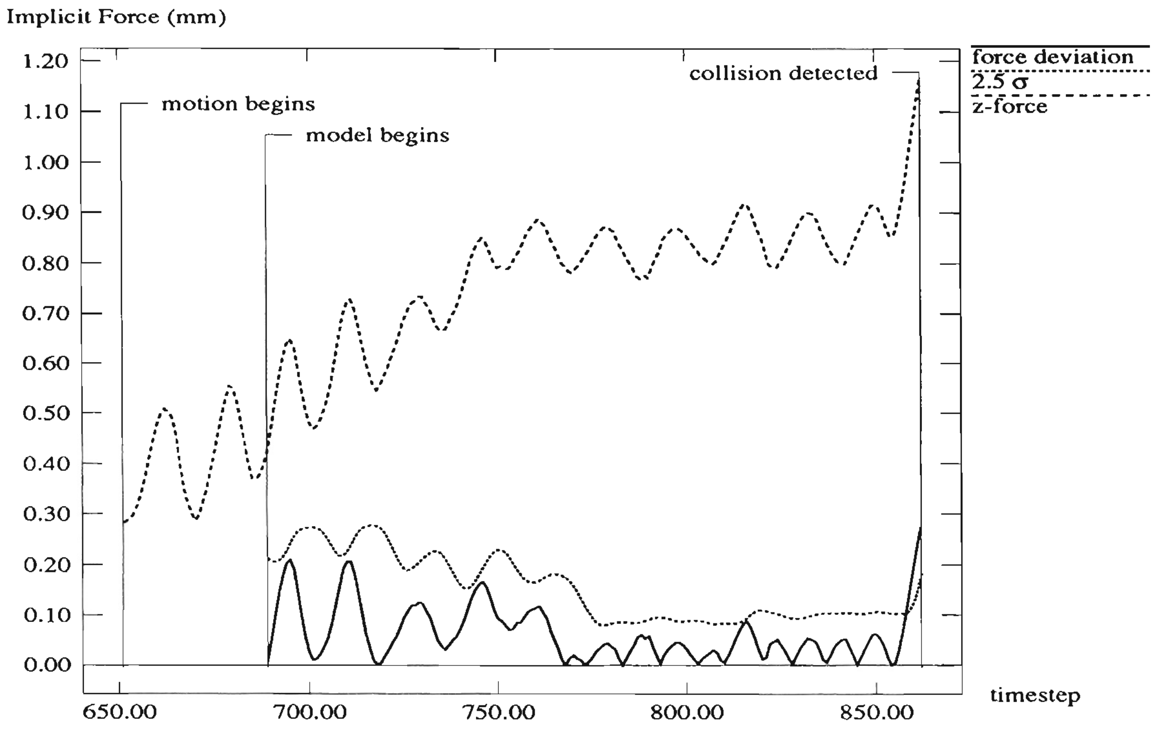


Figure 5.12: One Contact - Two Contact Collision

Figure 5.11 shows data from the first guarded move in the experiment. The collision is detected after the compliant wrist compresses 0.2mm (slightly over 2N force). The force deviates by over 3.0 standard deviations two other times during the move, but returns within the limits before 4 time steps.

Figure 5.12 shows data from the second guarded move in the experiment. The collision is detected after the wrist compresses approximately 0.275mm (under 2N force).

5.6 Limitations

There are certain situation where the combination of tolerance checking and collision detection will cause the system to identify an unsuccessful command execution as successful. One such situation is when an unexpected obstacle is next to a surface and smaller than the tolerance (see figure 5.13). The manipulator will stop against the obstacle, and continue on as if it had successfully collided with the surface. Another situation occurs during sliding motion when the surface friction is non-homogeneous. The manipulator can mistakenly identify a change in friction as a surface collision. When friction is very high, sliding motions may not be feasible. Finally, when approaching two surfaces simultaneously, the manipulator may contact the surface which is not the current goal first, which will cause the system to assume an error condition. Methods to reject these false detections have been developed[Donald 1988], but need to be implemented in a path planning algorithm, and thus are not applicable to the current situation. In this work, it is assumed that human intelligence in the path planning will help to avoid some of these situations. In the other situations, however, the compliance of the system and the size of the tolerance should both facilitate subsequent motions. It has been shown experimentally that in these situations, subsequent commands tend to correct for this type of problem.

5.7 Conclusions

Because the commands that use tolerance limits are susceptible to delay problems, it is likely that some post-processing of generated commands at the master site is needed.

The collision algorithm presented here appears to be effective for all of the experimentation we have done. It is simple, yet well suited to detect wanted signals from noisy sensor data. The delays caused by the algorithms are acceptable. Using this method of collision detection, the system is adaptable to a wide range of conditions, including changing surface conditions, velocity of impact, and tool mass. However, because of the statistical nature of the algorithm, it is still not 100% reliable. For instance, there is a limit to the amount of surface friction that the algorithm can reject, based upon the maximum threshold force. However, as the surface friction increases to this level, sliding along the surface becomes more difficult, to the point where sliding is not practical.

The use of the GuardVelocity() command will be illustrated in the next chapter.

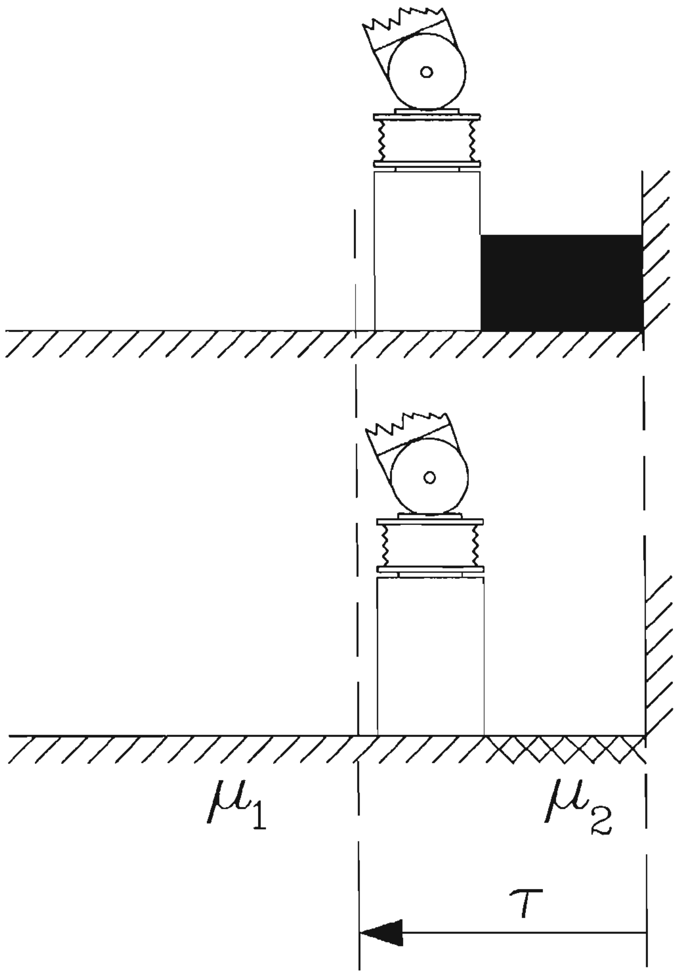


Figure 5.13: Undetectable Error Conditions

Chapter 6

Tool Usage

In order to increase the abilities of the teleprogramming system, the incorporation of tools into the remote system is explored. This chapter presents a control methodology for tools which does not increase the complexity of the human/system interaction, and does not increase the quantity of sensors needed at the remote site.

A robot manipulator is greatly limited in its strength to accuracy ratio. For a relatively low-strength robot to execute tasks which require more strength, powered tools are used in conjunction with the robot. The precision of the low-power robot is coupled with the strength of less-precise heavy tools. For this research, tools include an impact wrench, which delivers more torque than the robot, and a winch that has more lifting power than the payload capability of the robot.

Powered tools create extra degrees of freedom in the manipulator/tool system. A generalized system with redundant degrees of freedom creates a complex problem for the human operator to decide which degrees of freedom to control, or a complex computational problem if the decisions are made by the computer system. The manipulator/tool system, fortunately, is not a generalized system. If the tool has a natural axis of rotation or translation, the tool controls this degree of freedom, and the corresponding degree of freedom of the manipulator is made passive.

The tools used are themselves sensorless. They rely on the sensors of the manipulator (in this case, the instrumented compliant wrist sensor) for control feedback.

Because of this, the complexity of sensors is minimized. However, each tool must be controlled by the generalized sensor instead of a tool-specific sensor.

The implementation of two tools, a pneumatic-powered impact wrench and an electric winch are presented in this chapter. The use of the GuardVelocity() command, introduced in the previous chapter, is also presented here.

6.1 Background

Bolles and Paul used an electric screwdriver in their programmable assembly task [Bolles and Paul 1973]. Using the WAVE language [Paul 1977], the tool could be controlled by forces monitored by the manipulator. In a sense, the tool was controlled by guarded moves.

Although many industrial robots use tools, most are purely position controlled. The tool is treated as a separate device, and the only use of the manipulator is to move the tool into a required position and orientation. One notable exception is robots used for deburring, where the robot needs a sense of force from the deburring task in order to determine the feed rate for the robot motion [Asada and Liu 1991].

There is a need for tool usage by teleoperated systems to be addressed in terms of modern telerobotic research. One of the main concerns is to create a manipulator/tool system that is as easy to operate as the manipulator system by itself. A second issue is to minimize the sensory burden at the remote site. These issues are addressed here.

6.2 Tool Control

The most important consideration for controlling tools in a teleoperated system is not to increase the complexity of the human operator's task. However, a powered tool adds complexity to the system.

Powered tools add degrees of freedom to the manipulator/tool system. For example, an impact wrench attached to the end of the manipulator adds a rotational degree of freedom to the system (the natural rotation axis of the wrench). A winch, as another example, adds a translational degree of freedom in the direction of gravity. These extra degrees of freedom of the system are redundant. Choosing the directions to control and those to become passive is normally an optimization problem. Although straightforward, it is an iterative process, and therefore it is not often practical to solve this problem in real time. A generalized redundant system would be difficult to control in real time. However, the tool's functionality prescribes that control by the tool of its own natural degree of freedom is necessary for proper usage. This degree of freedom is automatically chosen as a direction to control.

When using a specific tool, a task frame can be assigned that aligns one cartesian direction of the frame with the natural axis of the tool, and this direction is controlled by the tool. For the robot manipulator, this direction becomes passive, and the system redundancy is removed. Because of the functionality of the powered tool, and the ability to use arbitrary task frames, control of the redundant manipulator/tool system is simplified.

The functionality of the tool is also important to the way the passive degree of freedom is treated. For example, the impact wrench is a force controlled device. The corresponding passive degree of freedom in the manipulator must be position controlled, so that it will not move. The winch, on the other hand, is a position controlled device. The corresponding degree of freedom in the manipulator is force controlled with a force preload, so that the manipulator will comply with the motion of the winch and keep the cable taut.

From the human operator's station, the control of the manipulator/tool system requires that a specific tool is chosen for use[1]. After this is accomplished, the operator can move the master arm around and see the tool working in the graphics environment. Motions by the human operator in the natural axis of the tool are used to control the tool. Some subtle changes are imposed on the feedback to the master arm. For example, control of the impact wrench is based on rotations about the z-axis of the master manipulator's tool frame. In order to prevent unwanted,

[1]Tool usage at the master site has not yet been implemented, and is beyond the scope of this research

accidental use of the wrench, a small amount of resistance to motion in this direction is programmed into the master arm controller. Therefore, only deliberate motions to control the tool are accepted as input. The winch control is designed similarly, with a further constraint that the velocity in the z-direction of the base frame is forced to be constant when deliberate motions in this direction are sensed, simulating the constant velocity of the winch. The human operator does not have to specifically control the tools; inputs of motion in the tool direction is all that is needed.

After the tool has been specified for use, commands are automatically generated in the same form as for non-tool actions. Motions, changes in contact states, guards, etc. are in the same form. The remote site, notified that a specific tool is in use, interprets these commands properly for the task.

6.3 Tool Sensing

The addition of a new sensor brings added complexity to the system. It must be calibrated and properly implemented, and the system will depend on the new sensor to work properly. Too many sensors become redundant, and sensor fusion techniques become necessary to correlate different sensor inputs. This is expensive in terms of time and computational power, as well as expensive in terms of the physical sensor devices. In order to simplify (and economize) the problem, all tool sensing feedback comes from the instrumented wrist.

The use of sensory inputs for tool control is similar to the implementation presented in the previous chapter. Guarded moves terminate properly when expected sensory events are encountered within the tolerance limits. Otherwise, motion is terminated in an error state. Guarded moves are generated automatically by the master system as required, in the same manner as a motion without tool usage. At the remote site, guarded moves may be interpreted slightly differently for tools. The tools may generate greater levels of noise than normal, and the remote site must be able to interpret sensor events from the noise. However, except for the noise level, the guarded moves are essentially identical for tool or non-tool motions.

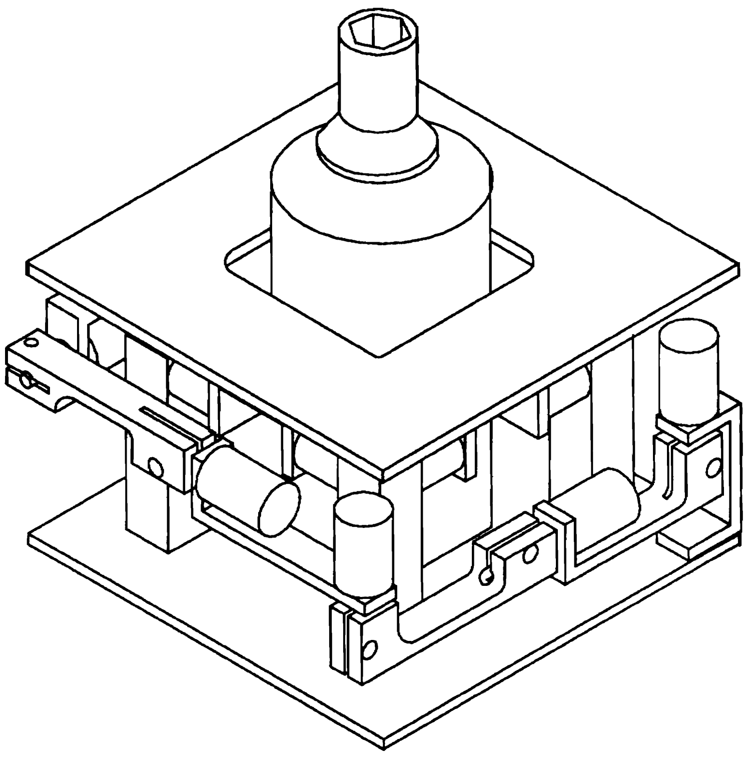


Figure 6.1: Wrist Outfitted With Impact Wrench

6.4 Impact Wrench

An impact wrench has advantages over a conventional wrench used by the robot. It can deliver much greater torque than the robot can by itself. The impact wrench is easier to use than the conventional wrench, because no complex movements are needed (however, most robot systems that use wrenches use electric or pneumatic wrenches which also do not require motion of the robot). Drawbacks to using the impact wrench include vibrations caused by the tool, and the need for a power source. The use of the impact wrench is characteristic of other tools the robot may use, such as drills and screwdrivers, which have the same natural axis of motion.

The impact wrench used for this research is a 3/8" drive pneumatic wrench. It accepts standard wrench sockets, but no provisions have been made in this system for changing sockets. Mounted internally on the wrist (see figure 6.1), it creates an extra degree of freedom with the same axis of rotation as the z-axis of the end effector frame

(T6). By specifying the wrench in the UseTool() command, rotation about the z-axis of T6 implies control of the wrench. The corresponding z-axis in the manipulator is position controlled with no specified motion, and thus acts as a locked joint (no motion allowed).

Inserting a bolt involves either sensing torques about the axis of rotation, or sensing velocity in the bolt feed direction. A high torque, or zero feed velocity, may indicate a jammed bolt or a properly seated bolt, so tolerances must be checked. Removing a bolt requires that the outward velocity be monitored. When the outward velocity goes to zero, the bolt is assumed to be fully extracted.

When the impact wrench is specified for use, the following parameters are automatically assumed:

- The z-rotation mode for the robot manipulator is set for position control. This assumes that the task frame is the tool frame of the manipulator ($T6$).
- Gains and maximum allowable forces/velocities are changed to reflect the increase in noise when the tool is on.
- Motions in the positive z-rotation direction are transformed to turn the impact wrench clockwise. Similarly, motions in the negative z-rotation direction turn the impact wrench on counterclockwise.

Use of the impact wrench is illustrated in the following example.

6.4.1 Sample Program Utilizing the Impact Wrench

The following is a sample program that is used to remove a bolt from the top of a box. Explanation and notes on the program will follow.

```
0.0    >> Execution Environment #0
0.1    UseTool(Wrench)
0.2    DefineVector(CP;<0.0,0.0,220.0>:EE)
0.3    DefineVector(X;<1.0,0.0,0.0>:EE)
0.4    DefineVector(Y;<0.0,1.0,0.0>:EE)
0.5    DefineTaskFrame(N:EE;CP;X;Y;?)
0.6    UseFrame(N)
0.7    AssignMode(P,P,P,P,P,P)
0.8    GuardForce(<0.0,0.0,-1.0>;<0.0,0.0,0.0>)
0.9    Move(5.0;<0.0,0.0,4.0>;<0.0,0.0,0.0>)
0.10   AssignMode(P,P,F,P,P,P)
0.11   Force(<0.0,0.0,1.0>;<0.0,0.0,0.0>)

1.0   >> Execution Environment #1
1.1   GuardVelocity(<0.0,0.0,1.0>;<0.0,0.0,0.0>)
1.2   Move(2.0;<0.0,0.0,0.0>;<0.0,0.0,-2.0>)
```

Execution environment #0 moves the impact wrench into contact with the bolt, via a guarded move. The first step is to define the impact wrench for use. A task frame is then created with origin at the tool tip (see figure 6.2), and the z-axis aligned with the rotation of the wrench. In the AssignMode() commands (0.7 and 0.11), note that the "P" in the sixth position is redundant, because the use of the impact wrench automatically implies this mode. A guard is set in order to stop the motion when the robot manipulator detects contact with the surface (command 0.8). The motion is specified in command 0.9. After contact is sensed, the post-motion commands (0.10 and 0.11) come into effect. Modes are switched to reflect the new contact state, and a force is exerted on the surface.

Execution environment #1 removes the nut from the bolt. A GuardVelocity() command is used to monitor the outward velocity of the nut. The motion indicated in the Move() command turns the impact wrench on, and the nut is removed.

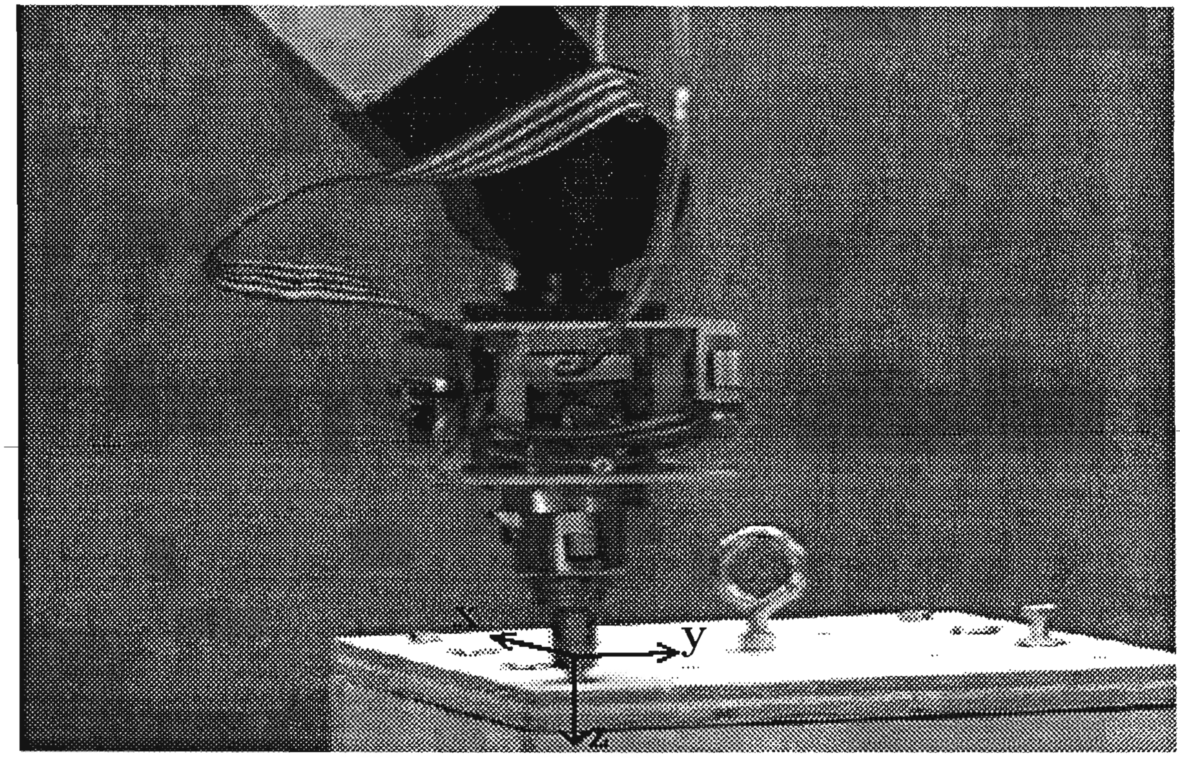


Figure 6.2: Impact Wrench Removing Bolt

6.4.2 Experimental Results

Figure 6.2 shows the robot-mounted impact wrench removing a bolt, as in the sample program above. The task frame coordinate system is indicated in the figure, with the origin at the tool tip.

The motion into contact is effective, as in the experiments from the previous chapter. Given the tolerance specifications of the current system, it is not easy to determine whether the socket has seated over the bolt, or merely rests on top of the bolt head. If this were a problem, tighter tolerances would have to be specified, or else exploratory procedures to determine proper seating would have to be implemented. Experiments have shown, however, that the socket does not have to seat properly for the bolt removal task to execute properly. In fact, the socket was properly seated after the motion into contact in less than half of the experimental runs, yet the removal of the bolt was still successful.

In order to sense when the bolt is fully removed, the GuardVelocity() command is used. Shown in figure 6.3 is the actual sensed velocity (raw sensor data) in the direction of the bolt axis as the bolt is removed, as well as the mean velocity based on 10 time steps of data[2]. The mean velocity model is used by the system for detection of sensor events because of the noise level of the raw sensor data. Note that the velocity model is not started until 1/2 second into the motion. This is to allow for motion transients which arise at the start of the motion to abate. In this run, for example, the socket did not initially seat over the head of the bolt. When the impact wrench is turned on, the socket moves down over the bolt head, as indicated by the initial positive velocity in the raw sensor data. As the robot complies with the force of the bolt, the feed rate of the bolt (determined by the bolt pitch) gives rise to a negative velocity in the robot. The mean velocity is monitored, and motion stops when the velocity returns to zero, indicating that the bolt is no longer feeding. A slight delay in sensing is caused as a result of the averaging of the velocity, but this does not adversely affect the task execution.

6.5 Winch

A winch can increase the load-carrying capacity of the robot manipulator. The need for power to offset the force of gravity on payloads, which usually accounts for a large fraction of the total power used by the robot, is eliminated. Because the payload capacity of a manipulator is usually inversely proportional to its accuracy, a robot with higher accuracy can be used for manipulation of heavy objects when a winch is used.

The winch is sensorless, and thus relies on the robot for sensing. The winch also adds a degree of freedom to the system, in essence adding a prismatic joint in the world z-coordinate. Motion in this direction is naturally controlled by the winch when the winch is specified with the UseTool() command. The corresponding degree

[2]Each time step is 20ms, the rate that the remote site computer communicates with the robot controller

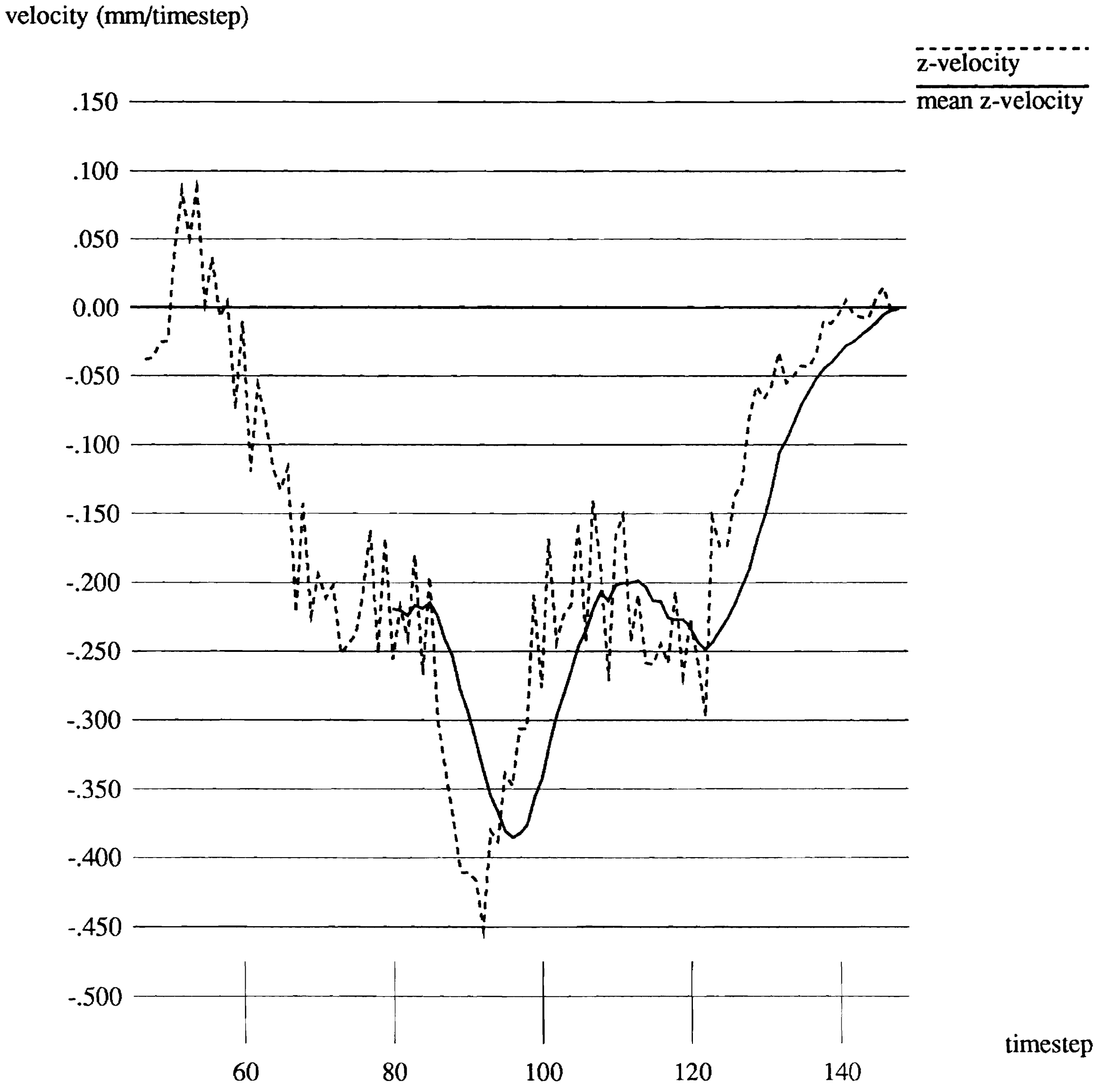


Figure 6.3: Velocity of Robot While Removing Bolt

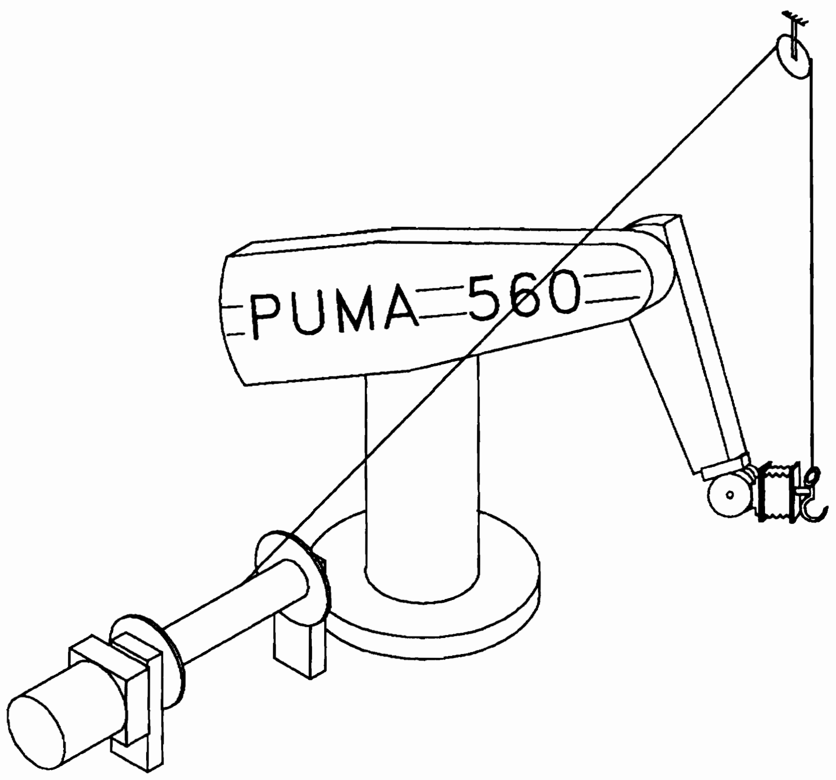


Figure 6.4: System Controlled Winch

of freedom in the manipulator is force controlled with a force preload to keep the winch cable taut.

The winch used for this research, illustrated in figure 6.4[3] is small, and not much more powerful than the robot itself[4]. However, knowledge gained about using the winch in conjunction with the robot can be applied to much more powerful winches. The winch operates at a fixed speed, and therefore the motions at the operator's station must be constrained to this motion when the operator is using the winch.

When the command to use the winch is parsed, the following parameters are automatically used:

- The z-direction mode for the robot manipulator is set for force control. The

[3]In figure 6.4, the winch, pulley, and robot are all fixed in the same base frame. Control of the winch/robot system where the winch and pulley are fixed in a frame that has motion relative to the base frame of the robot is a project beyond the scope of this research.

[4]The payload capacity of the winch is approximately 12 lbs.; the Puma robot payload is 5 lbs.

manipulator will then conform to forces in the z-direction.

- A force is set in the negative z-direction, to preload the winch cable (this keeps the cable taut).
- Gains and maximum allowable forces/velocities are changed to reflect the increase in noise associated with the constrained system.
- Motions in the positive z-direction are transformed to raise the winch. Similarly, motions in the negative z-direction lower the winch.

6.5.1 Sample Program Utilizing the Winch

The following is a sample program that is used to insert the winch hook into an eyebolt on the top of a box (see figure 6.5). Explanation and notes on the program will follow.

```
0.0   >> Execution Environment #0
0.1   UseTool(Winch)
0.2   DefineVector(X;<0.0,1.0,0.0>:KB)
0.3   DefineVector(Z;<0.0,0.0,1.0>:KB)
0.4   DefineTaskFrame(TF:EE;WST;X;?;Z)
0.5   UseFrame(TF)
0.6   AssignMode(P,P,F,P,P,P)
0.7   GuardVelocity(<0.0,0.0,1.0>;<0.0,0.0,0.0>)
0.8   Move(9.0;<0.0,0.0,-15.0>;<0.0,0.0,0.0>)

1.0   >> Execution Environment #1
1.1   DefineVector(C;<0.0,30.0,160.0>:EE)
1.2   DefineTaskFrame(PP:EE;C;X;?;Z)
1.3   UseFrame(PP)
1.4   Move(1.5;<0.0,0.0,1.0>;<0.0,-0.4,0.0>)
```

```
2.0  >> Execution Environment #2
2.1  Move(5.0;<-6.0,0.0,0.0>;<0.0,0.0,0.0>)

3.0  >> Execution Environment #3
3.1  AssignMode(P,F,F,P,P,P)
3.2  GuardForce(<2.0,0.0,0.0>;<0.0,0.0,0.0>)
3.3  Move(5.0;<-6.0,0.0,0.0>;<0.0,0.0,0.0>)

4.0  >> Execution Environment #4
4.1  DefineVector(C;<0.0,-52.0,180.0>:EE)
4.2  DefineTaskFrame(PP:KB;C;X;?;Z)
4.3  UseFrame(PP)
4.4  Move(1.5;<0.0,0.0,0.0>;<0.0,-1.0,0.0>)

5.0  >> Execution Environment #5
5.1  Move(9.0;<0.0,0.0,15.0>;<0.0,0.0,0.0>)
```

This simple set of instructions is typical of a program that would be automatically generated by the master system from simple motions of the master arm by the human operator. Absent are any exploratory motions that may be necessary to identify the location of the eye on top of the box.

Execution environment #0, above, moves the manipulator from free space into contact with the box (from (a) to (b) in figure 6.5). Command 0.1 informs the remote system that the winch is being used. This implicitly results in a force applied in the negative z-direction. Commands 0.2 to 0.5 define a task frame to be used by the control. It is important to note that when using the winch, the z-direction of the task frame must correspond to the z-direction in kinematic base coordinates. Because the system will be moving, however, the frame itself is defined to be dynamic in command 0.4, by specifying the frame name TF to be with respect to the dynamic frame EE. The AssignMode command (0.6) sets the hybrid control modes. The "F" in the third position in the command arguments is redundant, as the UseTool(Winch)

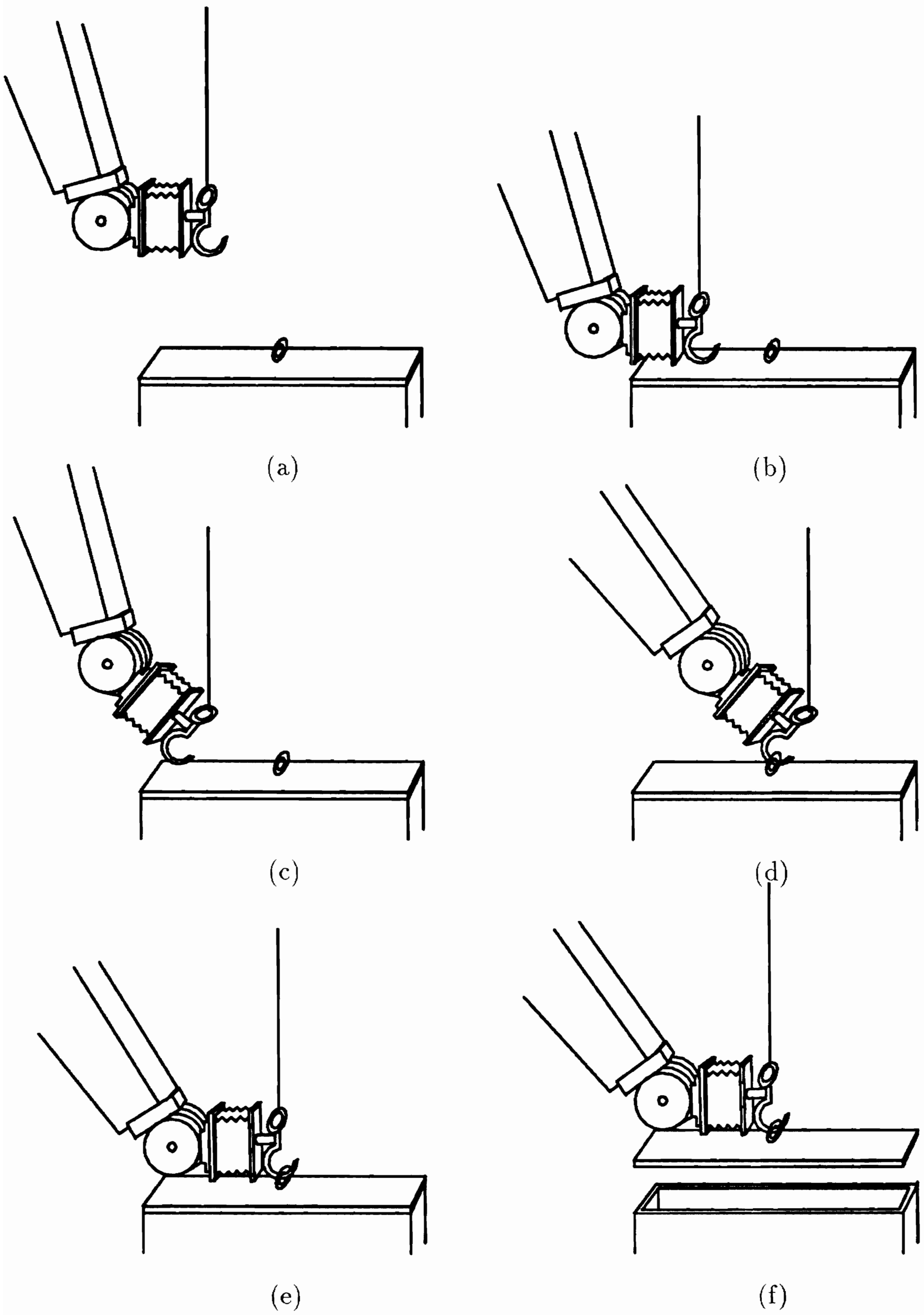


Figure 6.5: Winch Experiment

command will automatically set this.

The GuardVelocity command (0.7) is the guard for the motion into contact with the box. When the hook collides with the box, the downward velocity stops. The Move command (0.8) is purely a motion in the z-direction. It is controlled by motion of the winch, and the robot manipulator complies with this motion.

Execution environment #1 moves the hook slightly above the surface of the box and twists the hook, ready to move the hook through the eyebolt on the lid of the box (from (b) to (c) in figure 6.5). In order to rotate about the point that the winch cable connects to the hook, a new frame is created.

Execution environment #2 moves the hook to a point just in front of the eyebolt.

Execution environment #3 is a guarded move to mate the hook with the eyebolt (to (d) in figure 6.5). The AssignMode command (3.1) allows the hook to comply to the eyebolt, thereby allowing for inaccuracies in the positioning of the hook. The GuardForce command (3.2) will signal motion to stop when a force indicating the proper mating is sensed.

Execution environment #4 rotates the hook straight up (from (d) to (e) in figure 6.5). Here, a new task frame is created in order to rotate about the point of contact between the hook and the eye.

Finally, execution environment #5 lifts the top of the box off (from (e) to (f) in figure 6.5). Motion in the z-direction is controlled by the winch motion, and the robot manipulator complies with this motion.

6.5.2 Experimental Results

Figure 6.6 shows the winch lifting the top of the box, as in the sample program above, and the robot arm passively following the upward motion.

The robot is controlled in force mode in the z-direction with a force preload of approximately 2 lbs. on the winch cable. Any motion specified in the z-direction is assumed to have the velocity of the fixed-speed winch, and the winch is controlled up or down as specified. The robot complies adequately to this motion. All other

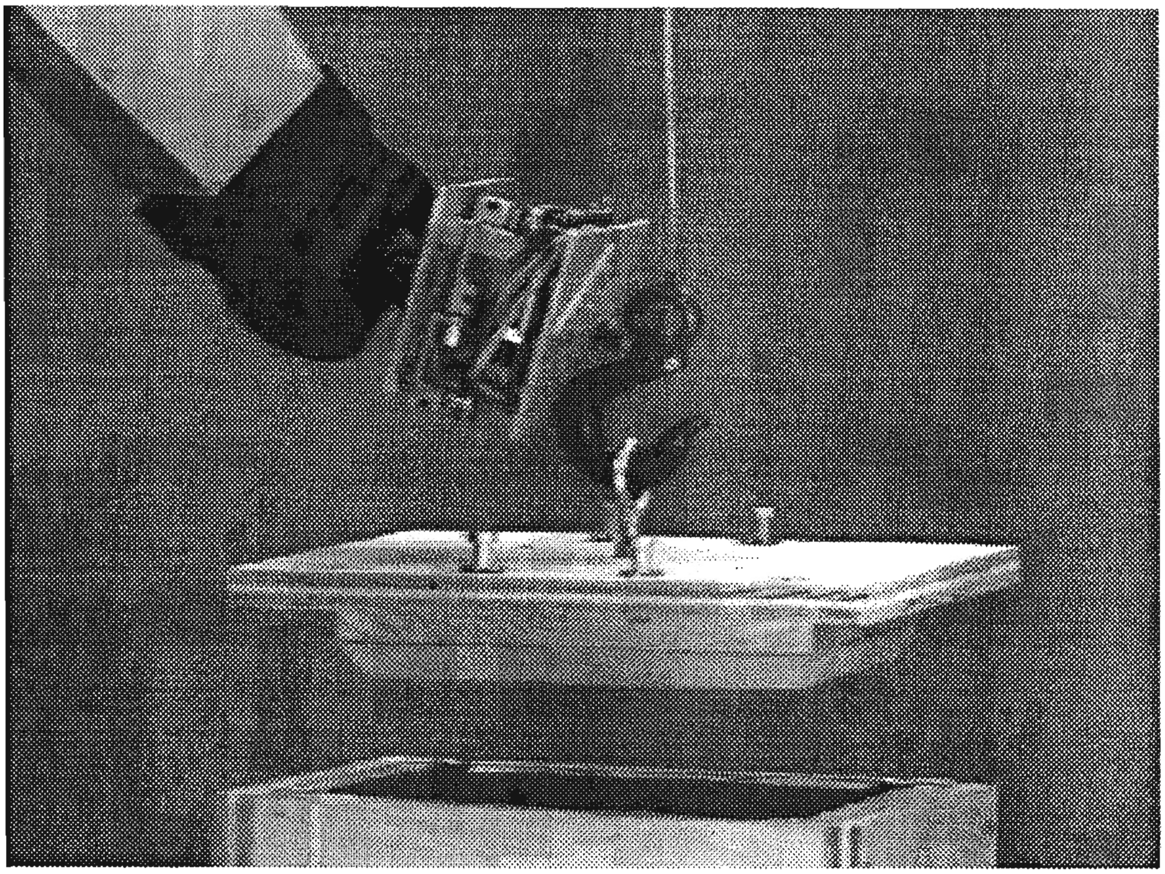


Figure 6.6: Winch Lifting Box Top

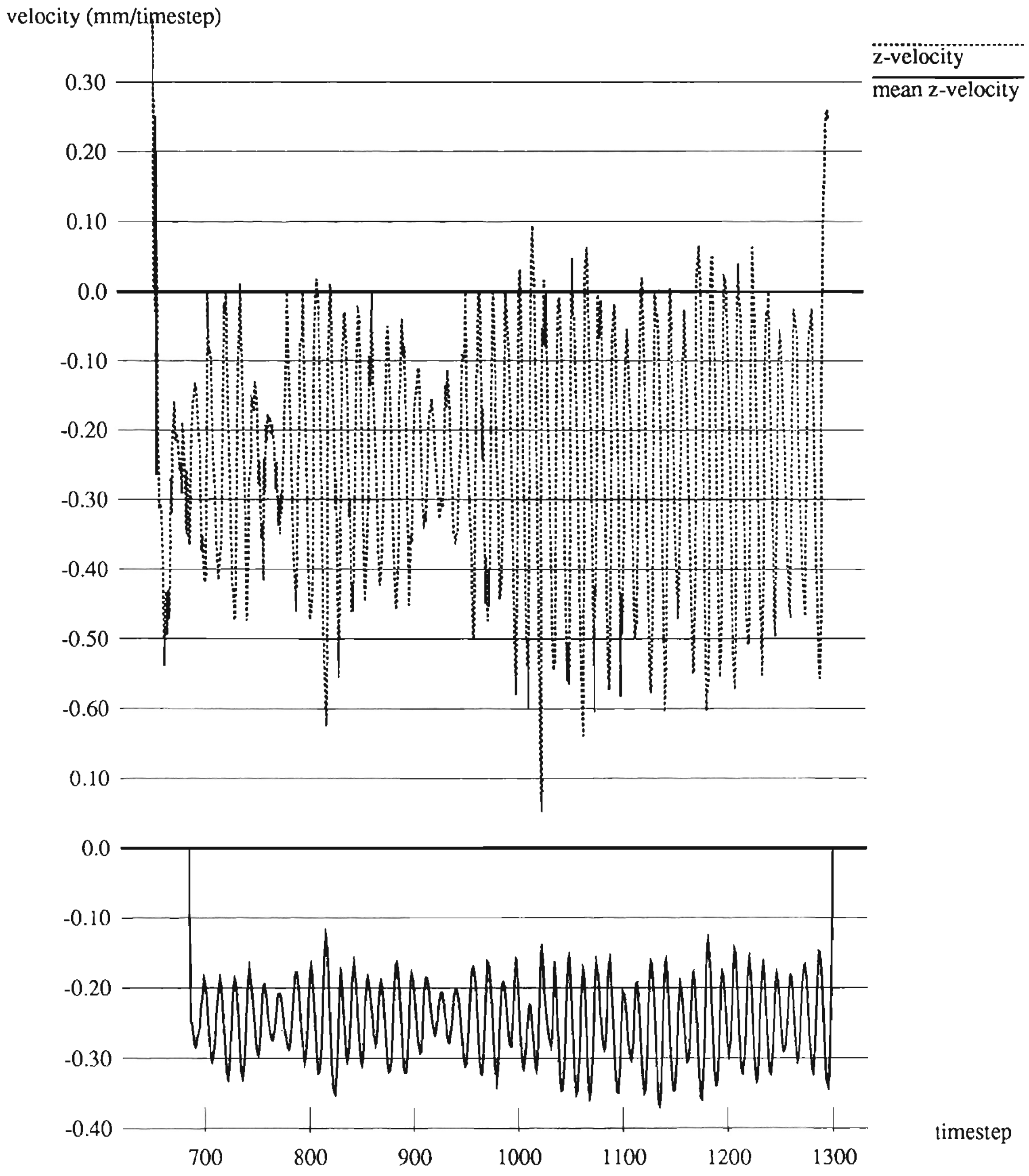


Figure 6.7: Hook Velocity Data for Guarded Move (a) to (b)

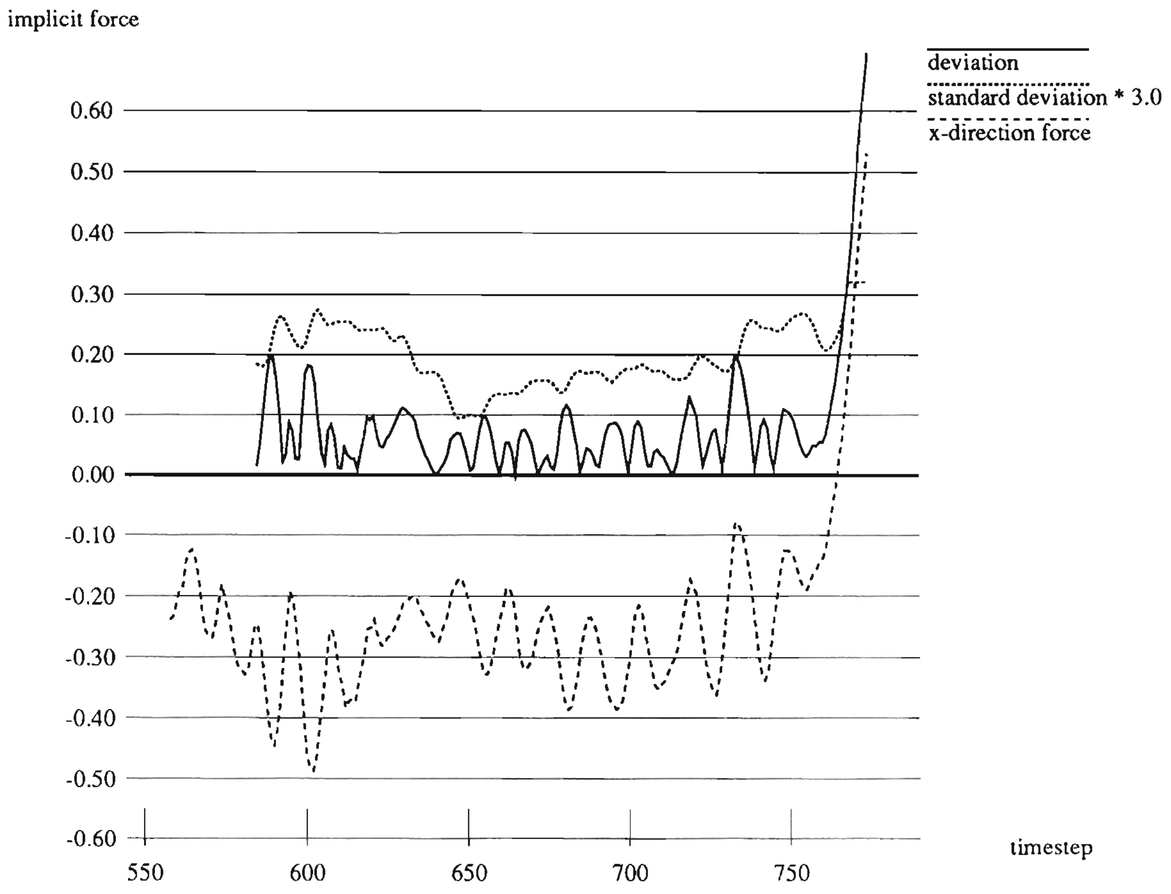


Figure 6.8: Hook Force Data for Guarded Move (c) to (d)

motion directions are controlled by the robot.

Sensing for the winch is more difficult than expected. Because the robot/tool system is constrained, internal forces are greater, and more noise is produced. It becomes more difficult to discern external event signals from sensor noise.

Figure 6.7 shows the actual sensed velocity (raw data) and mean velocity (computed over 10 time steps) of the hook in the z-direction for the motion commanded in execution environment #1 above (notice that there are two separate scales on the y-axis of the graph, in order to make the data easier to read). The mean velocity model is not initialized until 1/2 second into the move, to avoid any irregular or transient data when the move begins. The velocity data is very noisy, and has a periodic frequency of approximately 3.3 Hz, which can be attributed the natural frequency of the wrist/winch-cable mechanical system. The mean velocity data reduces the amplitude of this vibration and delays the signal by about 1/10 second. Using the mean velocity data, the change in velocity caused by the collision with the box is discerned easily. The delay of 1/10 second is acceptable.

Figure 6.8 shows the implicit force data for the move into contact with the eyebolt. The data shown is in the direction of the motion. Note that the average force in the motion before contact (before approximately 755 timesteps on the x-axis of the graph) is non-zero. This is caused by internal forces built up by the constraints of the system. The GuardForce command checks the deviation of the force data against the standard deviation model that is built up over the previous 30 time steps (.6 second). If the deviation of the force data is above 3 times the standard deviation for 6 consecutive time steps, a collision is detected. This rather stringent criteria is necessary in order to reject spurious, non-collision related data. It does, however, add slightly to the time delay in detecting collisions.

6.6 Conclusions

The methods presented here to control tools in a teleoperative system have the advantage that they do not increase the burden of work for the human operator. Also, because they do not require extra sensory input, the problems of sensor fusion and extra computational time needed to read and interpret additional signals are avoided, as well as avoiding the cost and implementation of another sensor in the system. This appears to be a successful implementation of tools in the teleprogramming environment.

A host of questions involving tool usage were raised during this research and not addressed. Among these are:

- What exploration techniques are necessary to find the bolt/eye/etc?

This question was not addressed in the examples presented here. Obviously, with a tolerance limit larger than the size of the bolt (for example), some exploration will be necessary. This leads to another question:

- What tolerances are necessary to be successful at removing a bolt, mating a hook and eye, etc.?

If the initial tolerance limits, defined by the accuracy of the initial imaging of the remote system, are not accurate enough to find or make contact with the bolt, some method of refining the accuracy of the model will be necessary. Efforts to use data collected at the remote site to refine the graphical model are currently in progress.

These first two questions can be more easily researched when the tool implementation at the master site is operational.

Other questions that may be asked include:

- Is the tool control implemented here too tool-specific?
- Have we chosen tools that are general enough to show usage characteristics of all powered tools?

The experiments presented here use only two powered tools. However, the methodology for tool usage appears to applicable to a wide range of tools that the robot can use. However, there are certain end effectors, namely grippers and hands, that add extra degrees of freedom to the system that can not replace coincident degrees of freedom of the robot manipulator.

Chapter 7

Conclusion

The development of the remote site system marks the successful conclusion of feasibility studies for the teleprogramming concept. We have experimentally demonstrated the effectiveness of coupling the human operator with a semi-autonomous manipulator system, in order to perform delay-invariant manipulation. The level of autonomy needed for the remote system is available with current computing and sensing technology.

Recent developments in battery technology, subsea acoustic communications, and subsea sonar/laser imaging systems indicate that teleprogramming can have immediate usage in shallow water where reliable communications can be guaranteed. With more robust communication signals[1], also within current technology, teleprogramming systems can be used in deeper waters. By eliminating the need for vehicle tethers, multiple autonomous underwater vehicles (AUVs) can operate in the same area without interference. The need for support ships is also diminished, as a communication link can be established from the AUV via an acoustic buoy to a remote location. Cost savings (in terms of the support vessel, tethers, etc.) could be significant. Furthermore, an AUV teleprogramming system and acoustic buoy could be deployed from a helicopter, greatly reducing the time before search/intervention procedures can begin. This is extremely beneficial in an emergency situation.

[1]Including internal error checking routines, multiple frequency/multiple transducer communications, etc

For shallow space applications, robust communication and accurate remote environment models would be available, and the teleprogramming paradigm is again directly applicable. Cost savings (in terms of human intervention, support operations, preparation time, etc.) will be considerable.

The remote system developed in this research, as well as the teleprogramming system as a whole, are delay-independent. The system performance remains constant with any time delays. When delays of less than one second occur, direct teleoperation techniques are applicable. However, the teleprogramming concept may offer advantages over direct teleoperation even in these situations. The system is also able to operate with delays of longer than 20 seconds. Future work will include functions to aid in these situations.

7.1 Contributions

The major contributions this research presents are outlined below:

1. A remote teleprogramming system has been developed, and demonstrated to be effective in interacting with a partially known environment.

2. Existing research in compliant instrumented wrist sensor-based hybrid control has been extended to use arbitrary task frames. The validity of remote task frames has been explored, with useful conclusions as to the distance a task frame can be located from the end of the robot.

3. Command execution has been developed for the remote system. Key elements are error detection (implemented by tolerance checks) and collision detection.

4. Tool usage by the teleprogramming system has been studied, with excellent results for using powered tools without adding to the complexity of the system and without creating an extra burden for the human operator.

5. As the operation of the remote system is completely independent of the time delay, the control structure used for teleprogramming is also available for direct

programming of the remote site robot. This is a useful tool for debugging the teleprogramming system, and also for directly controlling the robot for other purposes.

7.2 Future Work

Based upon the research outlined in this dissertation, various directions for future work, both immediate and long term, are presented here. Current work includes improvement of the master system performance, and implementation of operator aids to error recovery.

Remote system improvements. The remote system control can be improved with a more accurate model of the compliant wrist. The wrist stiffness matrix is currently modeled with only diagonal elements. The actual matrix is neither diagonal nor symmetrical. By using an improved model, control coupling can be minimized.

Master system improvements. As mentioned earlier, directions for improvement to the master system are discovered by the further development of the remote system. Post processing of automatically generated commands would lead to a more intelligent program, and eliminate the problem of small motion/large tolerance commands. Tool usage routines are also needed at the master site.

Command sequence replay. Common tasks, such as removing a bolt, are built up of a series of commands. Every bolt removing operation requires a similar set of commands, the only difference being possibly the approach vector and the length of the bolt. Such tasks can be pre-programmed as primitives, and used when the action to remove a bolt is indicated. Storing a sequence of commands which can be replayed every time a common task is needed can increase the human operator's productivity.

Practice mode. For hazardous tasks especially, the ability to practice tasks using the graphical model before any commands are sent to the remote site is desirable. Because the time delay is unimportant, it would be possible to either practice without command generation until an acceptable routine is worked out, or generate and store routines, and later send the best command set to the remote site.

Operator aids for error recovery. Information from the actual execution of commands at the remote site can be returned to the master site in order to improve the operator's understanding of error conditions. Actual motion and force information can be used by the operator to analyze the motions that lead up to an error. Further aid can be given to the operator when working with longer time delays. The sequence of commands generated by the operator after an error occurs at the remote site but before the operator is informed of an error are currently lost. If these commands are stored, it may be possible for the operator to recover from an error, and then replay the stored commands from the intervening time.

Real-time model refinement. The remote manipulator is often in contact with the remote environment. While in contact, the location of the manipulator and thus the environment is known to the tolerance of the manipulator system, which is likely to be more accurate than the initial visual (laser/sonar/etc.) model of the environment. This information can be used to refine the graphical model, and the overall accuracy of the system can be improved on-line.

Programs for manufacturing. The teleprogramming paradigm depends upon a human operator for decision making. The possibility of unexpected errors occurring, and error-prone recovery procedures in an unstructured environment proves the indispensability of the human operator. However, in a semi-structured manufacturing environment, for example, there may be only a few errors that occur frequently. By storing programs created by the human operator for the execution of such a task, including programs for correction of common errors, the teleprogramming system can be useful in a manufacturing environment. The operator would still be necessary to correct non-frequent, unprogrammed errors, but the mean time between these errors would be large enough that one operator could maintain a large number of manufacturing processes.

Bibliography

[An and Hollerbach 1987] An, C. H. and J. M. Hollerbach, Kinematic stability issues in force control of manipulators, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 897–903, Raleigh, NC, April 1987.

[Anderson and Spong 1992] Anderson, Robert J. and Mark W. Spong, Asymptotic stability for force reflecting teleoperation with time delay, *The International Journal of Robotics Research, 11,* April 1992.

[Asada and Liu 1991] Asada, Haruhiko and Sheng Liu, Experimental verification of human skill transfer to deburring robots, in *Proceedings of the Second International Symposium on Experimental Robotics,* Toulouse, France, 1991.

[Barber 1967] Barber, D. J., *MANTRAN: A Symbolic Language for Supervisory Control of a Remote Manipulator,* Master's thesis, Massachusetts Institute of Technology, 1967, S. M. Thesis.

[Bejczy et al. 1990] Bejczy, A. K., W. S. Kim, and S. C. Venema, The phantom robot: predictive displays for teleoperation with time delays, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 546–551, 1990.

[Bolles and Paul 1973] Bolles, R. and R. Paul, *The Use of Sensory Feedback in a Programmable Assembly System,* Technical Report, Stanford Artificial Intelligence Lab., October 1973, Memo 220.

[Campos 1992] Campos, Mario Fernando Montenegro, *Robotic Exploration of Material and Kinematic Properties of Objects,* PhD thesis, University of Pennsylvania, 1992.

[Donald 1988] Donald, Bruce R., A geometric approach to error detection and recovery for robot motion planning with uncertainty, *Artificial Intelligence, 37,* 223–271, 1988.

[Feigenbaum 1983] Feigenbaum, A. V., *Total Quality Control,* McGraw-Hill Book Company, New York, third edition, 1983.

[Ferrell 1965] Ferrell, William R., Remote manipulation with transmission delay, *IEEE Transactions on Human Factors in Electronics, HFE-6,* 24–32, 1965.

[Ferrell and Sheridan 1967] Ferrell, William R. and Thomas B. Sheridan, Supervisory control of remote manipulation, *IEEE Spectrum, 4,* 81–88, October 1967.

[Fisher and Mujtaba 1992] Fisher, W. D. and M. S. Mujtaba, A sufficient stability condition for hybrid position/force control, in *Proceedings of the IEEE International Conference on Robotics and Automation,* Nice, France, May 1992.

[Funda 1991] Funda, Janez, *Teleprogramming: Towards Delay-Invariant Remote Manipulation,* PhD thesis, University of Pennsylvania, 1991.

[Funda et al. 1992] Funda, Janez, Thomas S. Lindsay, and Richard P. Paul, Teleprogramming: toward delay-invariant remote manipulation, *Presence, 1,* 29–44, 1992.

[Goertz 1963] Goertz, R. C., Manipulators used for handling radioactive materials, in *Human Factors in Technology,* edited by E.M.Bennet, chapter 27, McGraw-Hill, 1963.

[Hirai et al. 1990] Hirai, Shigeoki, Tomomasa Sato, Toshihiro Matsui, and Masayoshi Kakikura, Integration of a task knowledge base and a cooperative maneuvering system for the telerobot "MEISTER", in *IEEE International Workshop on Intelligent Robots and Systems,* pp. 349–354, 1990.

[Hirzinger et al. 1989] Hirzinger, G., J. Heindl, and K. Landzettel, Predictive and knowledge-based telerobotic control concepts, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1768–1771, 1989.

[Inoue 1971] Inoue, H., Computer controlled bilateral manipulator, *Bulletin, Japan Society of Mechanical Engineering, 14,* 199–207, 1971.

[Lindsay and Paul 1991] Lindsay, Thomas and Richard P. Paul, *Design of a Tool-Surrounding Compliant Instrumented Wrist,* Technical Report MS-CIS-91-30, GRASP LAB 258, University of Pennsylvania, April 1991.

[Lindsay et al. 1991] Lindsay, Thomas, Janez Funda, and Richard P. Paul, Contact operations using an instrumented compliant wrist, in *Proceedings of the Second International Symposium on Experimental Robotics,* Toulouse, France, June 1991.

[Mason 1981] Mason, Matthew T., Compliance and force control for computer controlled manipulators, *IEEE Transactions on Systems, Man, and Cybernetics, SMC-11,* 418–432, June 1981.

[Niemeyer and Slotine 1991] Niemeyer, G. and J-J. E. Slotine, Stable adaptive teleoperation, in *Proceedings of the Second International Symposium on Experimental Robotics,* Toulouse, France, June 1991.

[Ogasawara et al. 1991] Ogasawara, T., K. Kitagaki, T. Suehiro, T. Hasegawa, and K. Takase, Model based implementation of manipulation systems with artificial skills, in *Proceedings of the Second International Symposium on Experimental Robotics,* Toulouse, France, June 1991.

[Paul 1977] Paul, R. P. C., Wave: a model based language for manipulator control, *The Industrial Robot, 4,* 10–17, 1977.

[Paul and Shimano 1976] Paul, Richard and Bruce Shimano, Compliance and control, in *Joint Automatic Control Conference,* 1976.

[Paul et al. 1990] Paul, Richard P., Janez Funda, Thierry Simeon, and Thomas Lindsay, Teleprogramming for autonomous underwater manipulation systems, in *Intervention '90,* pp. 91–95, The Marine Technology Society, June 1990.

[Paul et al. 1992] Paul, Richard, Thomas Lindsay, and Craig Sayers, Time delay insensitive teleoperation, in *IROS 1992,* 1992.

[Raibert and Craig 1981] Raibert, M. H. and J. J. Craig, Hybrid position/force control of manipulators, *ASME Journal of Dynamic Systems, Measurement, and Control, 103,* 126–133, 1981.

[Roberts et al. 1985] Roberts, Randall K., R. P. Paul, and Benjamin M. Hillberry, The effect of wrist force sensor stiffness on the control of robot manipulators, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 269–274, April 1985.

[Salganicoff 1992] Salganicoff, Marcos, *Visually-Driven Grasp Planning,* PhD thesis proposal, University of Pennsylvania, 1992.

[Shutter and Brussel 1988] Shutter, J. De and H. Van Brussel, Compliant robot motion, *International Journal of Robotics Research, 7,* August 1988.

[Sinha 1991] Sinha, Pramath Raj, *Robotic Exploration Of Surfaces And Its Application To Legged Locomotion,* PhD thesis, University of Pennsylvania, 1991.

[Volpe and Khosla 1991] Volpe, Richard and Pradeep Khosla, Experimental verification of a strategy for impact control, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1854–1860, 1991.

[Wav 1992] Waves: the international ocean technology news magazine, March/April 1992, page 6.

[Whitney 1982] Whitney, Daniel E., Quasi-static assembly for compliantly supported rigid parts, *ASME Journal of Dynamic Systems, Measurement, and Control, 104,* 65–77, March 1982.

[Whitney 1985] Whitney, D. E., Historical perspective and state of the art in robot force control, in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 262–268, 1985.

[Will and Grossman 1975] Will, Peter M. and David D. Grossman, An experimental system for computer controlled mechanical assembly, *IEEE Transactions on Computers, C-24,* 879–888, 1975.

[Xu 1989] Xu, Yangsheng, *Compliant wrist design and hybrid position/force control of robot manipulators,* PhD thesis, University of Pennsylvania, 1989.

[Xu and Paul 1988] Xu, Y. and R. Paul, On position compensation and force control stability of a robot with a compliant wrist, in *Proceedings of the IEEE Conference on Robotics and Automation,* pp. 1173–1178, 1988.

[Yoshikawa 1990] Yoshikawa, Tsuneo, *Foundations of Robotics,* MIT Press, 1990.

[Zhang 1986] Zhang, Hong, *Design and Implementation of a Robotic Force and Motion Server,* PhD thesis, Purdue University, 1986.