Whole-Arm Manipulation: Kinematics and Control

by

Brian Scott Eberman

S.B (1986) Mechanical Engineering
Massachusetts Institute of Technology

Submitted in Partial Fulfillment of the Requirements of the Degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

January, 1989

Signature of Author ______________________________

Department of Mechanical Engineering
January, 1989

Certified by ______________________________

Kenneth Salisbury
Thesis Supervisor

Accepted by ______________________________

Ain A. Sonin
Chairman, Department Committee on Graduate Students

Whole-Arm Manipulation: Kinematics and Control

by

Brian S. Eberman

Submitted to the Department of Mechanical Engineering on January 20, 1989 in partial fulfillment of the requirements for the Degree of Master of Science in Mechanical Engineering

Abstract. This thesis explores the kinematics and control issues associated with an approach to manipulation that employs all of the available surfaces of a robot to interact with the workspace. This thesis explores manipulations of this type which we term whole-arm manipulation (WAM). A classification of WAM tasks into pushing, searching, enclosure, and exclusion is presented and examples of each are given. A technique for describing the task kinematics and compliance in terms of compliant line motions is presented. Compliance control requirements are discussed in light of these task requirements.

Implementation of the kinematic solutions and control for whole-arm tasks is described for the MIT-WAM robot. The inverse kinematic problem of placing the last link of the robot along a desired line is derived and the resulting transformations are applied to a maximum path deviation algorithm to plan manipulator motions. The endtip cartesian kinematics is derived and applied to the problem of sensing the location and magnitude of a contact force. Experimental results are presented for an approach to contact which uses joint torque information to determine contacts. Finally, we describe an application of the system to the whole-arm task of searching for a set of objects.

, ,

Thesis Supervisor: Dr. Kenneth Salisbury
Research Scientist
Laboratory of Artificial Intelligence

Acknowledgments

This thesis describes research conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Office of Naval Research University Initiative Program under the Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

I would like to thank the members of Ken Salisbury's research group. Bill Townsend designed and built the robot on which all of the work in this thesis was performed. Dave Brock provided many illuminating discussions into the wee hours of the night. Ben Paul provided an initial introduction to the laboratory and visual aids for late night studying. A new addition to the group, Mike Levin, helped to renew my efforts and took care of many of the final details that were needed to finish this project.

The assistance of a number of undergraduates was essential to the completion of this thesis. Sam Hou, Beth Green, Henry Dotterer, and Kimo Tam helped me to build all of the electrical systems that went into making the robot run. Kent McCord's work in the machine shop and in the laboratory produced many of the components in the robot system.

I would like to thank the members of the general laboratory community that were so helpful. Sundar Narasimhan guided me through the intricacies of UNIX and the Condor. Enlightening conversations with Anita Flynn, Al Ward, Eric Vaaler, Neil Singer, among others assisted me in robotics, electrical design, and in the machine shop.

Ken Salisbury has provided the guidance, support, and encouragement that has made this thesis possible. Innumerable discussions on robotics and control shaped the ideas in this thesis and helped me grow as a researcher.

The love and support of my parents, Paul and Kay Eberman, has been invaluable. They have offered words of wisdom and encouragement. Finally, Margaret Covert, my fiancee, has provided encouragement, love, help, and companionship.

Contents

Chapter 1

Introduction

1.1 Present Robots

Robot tasks can be grouped into two general types based on how the task changes the environment. A non-contact task, such as spray painting and welding, involves manipulating an end-effector to cause an effect at a distance. Machining, grinding, and assembly use the end-effector to exert forces directly on the environment and are classified as contact tasks. Contact and non-contact tasks have been studied extensively in the literature and similarities in manipulator designs and control strategies have resulted.

In a typical non-contact task, the manipulator is required to move a heavy end-effector (e.g. a welding gun or spray painting tool) through its workspace precisely and rapidly. The driving robot design goals are the repeatibility, accuracy, and speed of the end-effector. Resulting designs have tended to be constructed of rigid links driven by powerful actuators. Design criterion for this type of manipulator can be found in Youcef-Toumi [Youcef-Toumi 85] and Asada [Asada 84]. Excellent treatments of end-effector control can be found in Asada and Slotine [Asada and Slotine 86], and Craig [Craig 87].

Contact tasks can be subdivided once more into two sub-categories: Tasks which modify the contacted object's geometry, such as machining and grinding, and tasks which move objects into new configurations (e.g. assembly). The first type of task requires a rigid manipulator capable of exerting large forces on the workpiece, powerful actuators and rigid links are the resulting design. The second function requires a robot that can interact with the environment with a controllable interaction force. Manipulator compliance and force control, are techniques for controlling the interaction with feedback. Only recently have manipulators been designed specifically for force control.

Asada [Asada, Kanade 81] and Youcef-Toumi [Youcef-Toumi 85] constructed direct-drive arms for force control to eliminate transmission backlash, a source of limit cycling in force and position controlled manipulators. An [An 86], using the Asada direct-drive arm, successfully implemented algorithms to control the interaction force at the end-effector. Townsend [Townsend 88] (the MIT-WAM arm) and DiPietro [DiPietro 88] built cable driven robots to reduce backlash and friction and consequently improve the performance of force control systems. One of the objects of this thesis is to demonstrate the success of the MIT-WAM cable driven arm in achieving active compliance.

Compliant control has also been studied in the field of hand manipulation. Salisbury [Salisbury 84b] and Jacobsen [Jacobsen 84], [Jacobsen 86] describe the design of complex hands. Chiu [Chiu 85] and Narasimhan [Narasimhan 87] discuss compliance control and kinematics issues for multiple point grasps. Research in dexterous hands has mostly focused on grasps involving contact between each of the fingers and the manipulated object at the fingertips. Fingertip grasps enable precise control of the object and adroit object manipulation, but limit the amount of force which can be exerted on the workpiece. Inner-link grasps, or power grasps, where contacts occur at more then one point along a link are more stable in the face of environmental disturbances and can exert higher forces on a grasped object. However, they are geometrically more complicated and consequently more difficult to analyze.

Future robotic tasks will require more than just end-effector or endtip manipulation. Robots are beginning to enter the construction industry, the service industries, light manufacturing, and assisting in space and underwater environments. Robots might be used to sort baggage for the airlines, to move packages at UPS and the post office, to inspect and repair nuclear facilities, to paint buildings, and clean floors. All of these tasks take place in an unstructured environment. In contrast to the factory floor, all of the objects in the manipulators workspace are not under human control. People, animals, robots and other rapidly moving objects may be present. Objects which are to be manipulated are not precisely located. A manipulator, or autonomous vehicle, that is going to work in this environment, must be able to adapt to the changing environment and conform to surfaces when the inevitable collision occurs.

To operate in an unstructured environment, robots need to possess a number of important features. They need a variety of real-time sensing capabilities including contact sensing, tactile sensing, force sensing, and real-time vision systems. Collisions will occur, and the manipulator must be designed to survive the collisions and not damage the object with which it has collided. The machine will have to manipulate a wide range of object geometries, sizes, and weights. This will require a variety of techniques for manipulating the objects. Lastly a flexible, adaptable control algorithm and programming structure will be needed to control the robot.

1.2 Thesis Overview

This thesis will explore the concept of whole-arm manipulation (WAM) for control of robots in an unstructured environment. First put forth by Salisbury [Salisbury 87] and elaborated on by Salisbury et al. [Salisbury 88] and by Townsend [Townsend 88], a WAM manipulates objects with all of its moving surfaces. A set of geometric and kinematic design criteria for this type of manipulator is given in [Townsend 88] and [Salisbury 88] and is briefly reviewed in chapter 2. Whole-arm manipulation requires robust force control and consequently places constraints on robot design. These design issues are reviewed in chapter 3.

Chapter 2 describes whole-arm manipulation in detail and describes a possible taxonomy of tasks that can be accomplished without an end-effector. The concept of "line manipulation" is introduced and its kinematic relationships derived in order to aid task specification.

Chapter 3 discusses control issues for unstructured environments. Present force control techniques are reviewed briefly. The requirements for a clean mechanical system required for force control are also reviewed. Techniques for estimating and controlling the nonlinear effects in the mechanism and motor controllers is presented. Lastly, an overview of the control approach used in our implementation of whole-arm control is described.

Throughout this thesis, the emphasis is on actual implementation and experiment. The last chapters, then, are a case study of whole-arm kinematics and control. Chapter 4 describes the kinematic issues for whole-arm manipulation as applied to our manipulator. After a brief description of the arm's geometry, the kinematic relationship for the endtip location in cartesian space as well as the kinematic relationship between the line formed by the last link and lines specified in cartesian space are derived. Lastly, the generation of joint trajectory motions from cartesian or line motions is detailed.

Chapter 5 reports on the implementation of the stiffness control system described in Chapter 3. The computer hardware and software used for the real-time system is described. Finally the effects of the nonlinear feedforward compensation is discussed.

Chapter 6 describes manipulator performance results. The stiffness of the MIT-WAM is presented. The bandwidth of the manipulator and the range of controllable forces is described. Lastly, application of the line kinematics and stiffness to whole-arm tasks is presented.

Chapter 7 reviews the major points of the thesis and presents directions for future research.

Chapter 2

WAM Tasks and Kinematics

2.1 Whole-Arm Manipulation (WAM)

The ideas in whole-arm manipulation are derived from the requirements for manipulation in an unstructured environment. Unlike conventional robots which focus on endtip manipulation, a robot designed for whole-arm manipulation (WAM) is designed to contact and manipulate the environment with all of its moving surfaces. A robot designed for whole arm manipulation should have certain characteristics. Townsend [Townsend 88] lists these as:

- The geometry of the links should be clean and free of dangling wires and protrusions which could snag on the environment.
- The link surface should be covered with a compliant surface to minimize the impact forces. This covering should have Coulomb like frictional properties to aid in manipulation.

- The links should be long and slender so that the range of objects which can be grasped is maximized. Long, slender links (links with a high aspect ratio) occupy the smallest amount of the manipulator's workspace.

- The manipulator should be able to exert a wide range of stable forces on the environment. (Conditions for this are reviewed in chapter 3).

- The design should be robust and of minimum complexity.

A robot constructed in this way can use all of its surfaces to manipulate objects. This increases the range of object sizes and weights which can be accommodated. Generally, substantially heavier objects can be moved by pushing with the edges of a link then can be moved by lifting with the end-effector. Since the torques on the manipulator joints caused by an external force vary proportionately with the distance from the joints, it is clear that much heavier objects can be moved near the joints than at the end-effector.

Control in an unstructured environment requires the ability to detect the location of contact. In view of the requirement of minimum complexity, we wish to use the manipulator as a contact/force sensor. In certain configurations, the contact forces and the contact location can be inferred from measuring the joint torques. This is examined in greater detail in chapter 4. The joint torques can be determined by either using torque sensors or by using a low friction transmission and a high-quality actuator system.

Given a manipulator designed in this way, the set of tasks which can be accomplished must be examined. There are numerous examples in human activity and human controlled equipment where manipulation using inner link contact is important. Surfaces can be used to gain leverage for an arm and increase the capacity of a strong end-effector to move objects [West 87]. Humans use their arms to cradle heavy loads, use grasps between their arms and their bodies to constrain large objects, and use their shoulders and sides of their arms to push on large objects. A back-hoe uses the sides of its shovel to smooth loads of earth. A mason uses a trowel to push and smooth cement. Painting and sanding, although accomplished with an end-effector, involve using the brush surface to push paint along a surface. All of these tasks are examples of whole-arm manipulation.

The variety of WAM tasks can be grouped into four main categories: pushing, searching, enclosure, and exclusion. Although this taxonomy is neither complete nor mutually exclusive, it does encompass a variety of useful tasks. Each of these tasks can be described by a kinematic relationship between the programmer's space of interest and joint space by specifying the location, orientation, and motion of lines. This description can be applied to one or many manipulators acting in concert.

2.2 Taxonomy of Tasks

In this section a set of four basic tasks is described to illustrate the types of tasks that can be accomplished with the entire arm. Although this set of tasks is not necessarily complete, it does provide a useful basis for developing a task level description of arm motion.

2.2.1 Pushing

Pushing is defined here as acting on an object with a force or set of forces to cause the object to move, and is distinguished from grasping in that the object may not be fully constrained by the manipulator contact forces. Many everyday tasks are accomplished through pushing. Backhoes, bull-dozers, and most construction equipment work through pushing. Package sorting and handling, moving heavy blocks, and straightening a deck of cards are all accomplished through pushing.

Pushing can be used to manipulate objects that would otherwise be to heavy to lift. By using kinematic constraints in the environment, objects can be aligned and uncertainty in their orientation can be reduced. Consider the following example derived from Mason [Mason and Salisbury 85]: A block is to be oriented with its long edge along a fence as in figure 2.1. The fence translates in a direction, the *line of pushing*, until contact is made with the block. At this point, the motion of the block can be determined by constructing the contact normal, at the contact location, and lines at an angle from the contact normal equal to the arc-tangent of the coefficient of friction. The sector between the two lines is called the *cone of friction*. If this sector passes entirely on one side of the center of friction (as described by [Mason and Salisbury 85]), the direction in which the block will rotate is given by the moment of the sector about the center of friction.

In the case where the center of friction lies within the cone of friction, the line of pushing is used to determine the direction of rotation. If the line of pushing passes to the left of the center of friction, the block rotates clockwise. Unless the center of friction

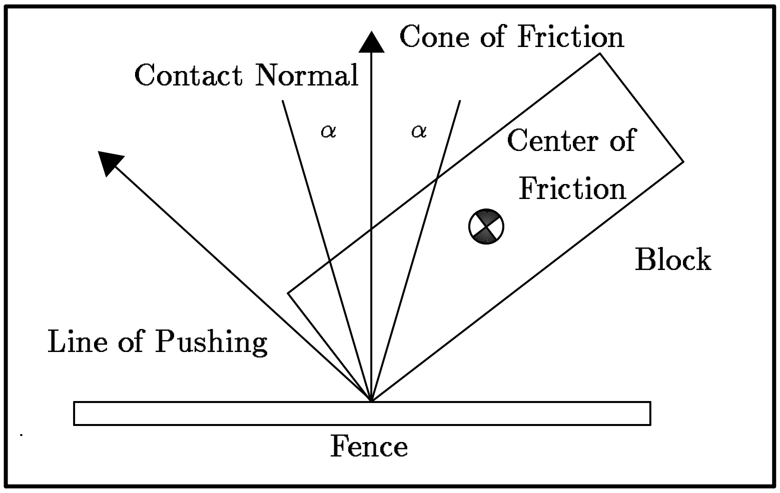

Figure 2.1: **Initial Contract with a Fence.** If the cone of friction passes entirely to the left of the center of friction the block will rotate clockwise. If the center of friction lies within the cone of friction the line of pushing determines the direction of rotation. The angle α is equal to $tan^{-1}(\mu)$.

both lies along the line of pushing and lies within the friction cone, the block rotates and comes into contact with the fence. This contact is stable if the lines of pushing through the two touching block corners pass on either side of the center of friction (see figure 2.2.

If gravity is the only force acting on the object in the vertical direction and if the coefficient of friction is uniform between the object being pushed and the support surface, the center of friction is at the object's center of gravity. The addition of forces in the vertical direction, other than gravity, shifts the center of friction toward the applied force. More detail can be found in [Mason and Salisbury 85].

The block can now be in one of four states corresponding to each of the possible edges. To select the long edge, the fence can be translated in the direction shown in figure 2.3. If the coefficient of friction is sufficiently high, the block will fall and align

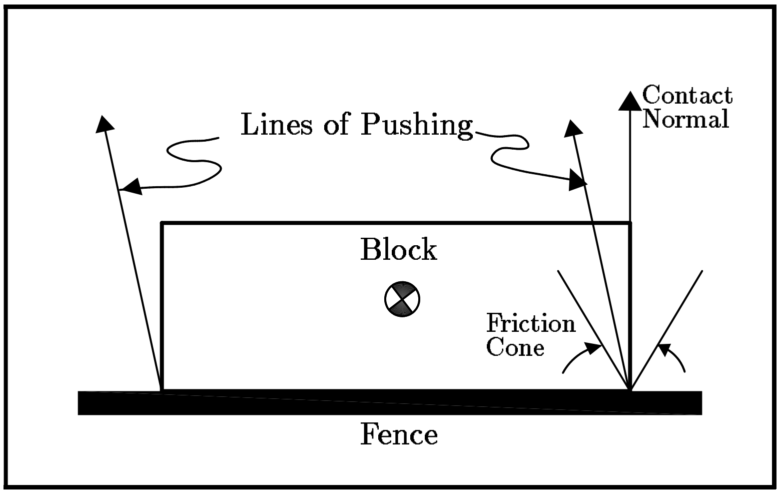

Figure 2.2: **Stability of Pushing.** The figure above has lines of pushing on both sides of the center of friction. This is a stable push.

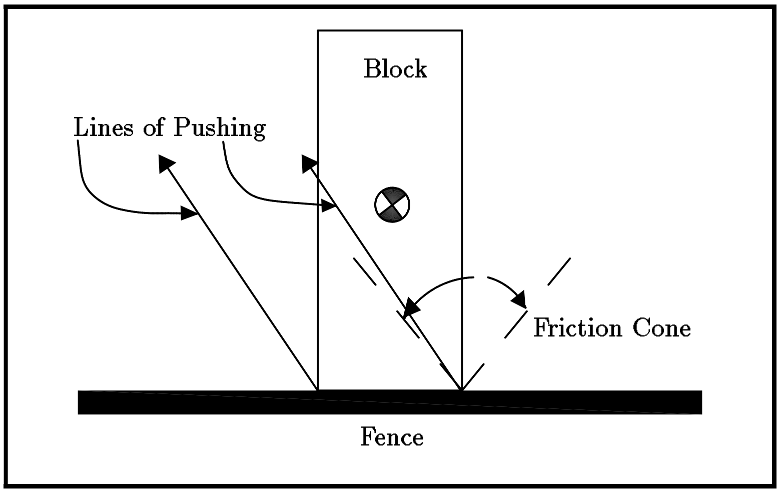

Figure 2.3: **Unstable Push.** Both lines of pushing pass to the left of the object's center of friction. The object will rotate clockwise until contact with the fence again is made. Note that the center of friction lies within the cone friction for the right line of pushing. If this condition is not satisfied, the block will slip along the fence and will not rotate.

its long edge with the fence; however, if the coefficient of friction is too low, the block will slip along the fence and the procedure will fail. However, if the fence is the edge of a manipulator, and if there is a method available for determining which configuration the block is in, the manipulator can simply be actively repositioned approximately along the correct edge and the block alignment procedure repeated. This procedure can be repeated for a number of blocks and, with the addition of the ability to sense contact between the manipulator and another object, can be used to arrange a set of blocks in a regular pattern without grasping the blocks.

The procedure is robust to errors in the location of the block and calibration errors. If the last link contacts the block in such a way that when the block finishes rotating the entire edge of the block lies along the last link, the manipulation will succeed. Depending on the length of the link and the length of the block, there could be a large region of successful contact locations. In addition to the robustness of the procedure, the link may be able to align multiple blocks with a single push.

The analysis of pushing assumed that we could translate a fence or line in a given direction. The structure of an actual manipulator will impose constraints on the motion of the contact points. For a two-degree-of-freedom manipulator, pushing is accomplished by translating the line which passes through the last link. In order to translate this line in the direction perpendicular to the line, the second link must move along the line. Thus, the actual line of pushing is not aligned with the direction of motion of the line, instead the true direction is shown by the path in figure 2.4. The object moves along this path if the coefficient of friction between the block and the manipulator is large enough

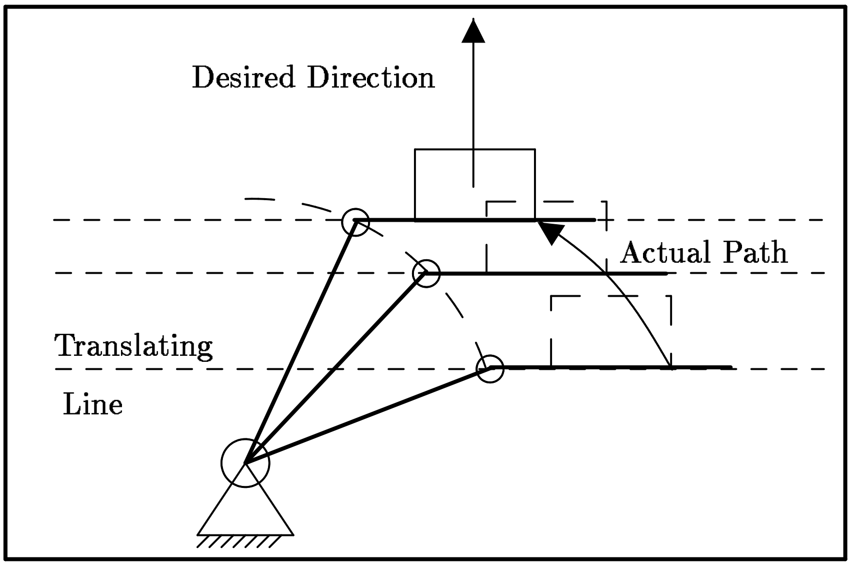

Figure 2.4: **Translation of a Block with a Robot Link**. A two degree-of-freedom robot cannot push a block in a straight line.

to prevent slipping. To remedy this problem, either the translation must be accepted or the line can be rotated as it translates, and a surface at the end of the path can be used to align the block. An additional solution to the problem is to allow the block to slip along the last link by using a low friction coating on the manipulator; however, by allowing slippage the location of the block along the manipulator becomes more difficult to determine.

For a spatial manipulator with more than two freedoms, the problem can be overcome. In the example shown in figure 2.5, a four degree-of-freedom manipulator is being used to push a block. The line of pushing can be translated in the desired direction by re-orienting the last link along the face of the block. If the location of the manipulator's last link is projected into the plane of support, it can be seen that the projection of the

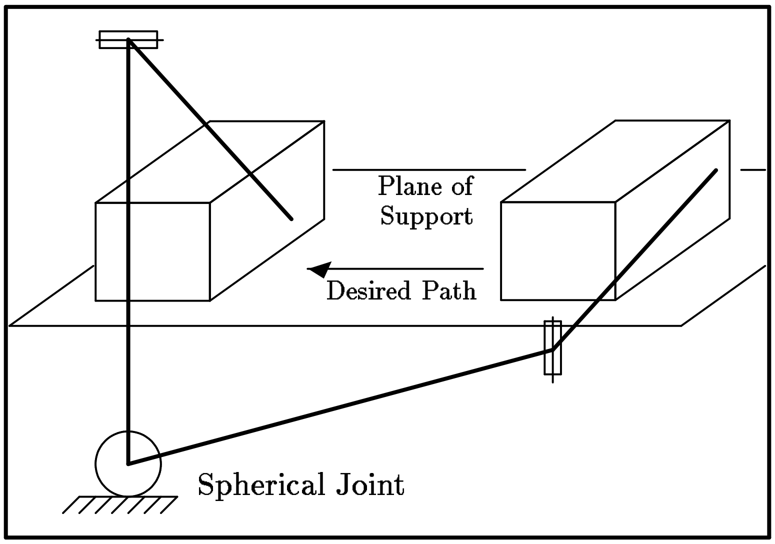

Figure 2.5: **Pushing a block with a four degree-of-freedom robot**. As the manipulator pushes on the object in the desired direction, the orientation of the last link changes along the face being pushed.

last link translates in the direction of the desired line of pushing.

2.2.2 Search

Search is defined here as moving a compliant kinematic structure through the workspace in order to detect objects by colliding with them. In searching, as opposed to pushing, the manipulator must not cause the object to move when the contact is made or while the object is being explored. Given a manipulator, limits to the size and weight of objects which can be searched can thus be defined.

In order to perform a search, the manipulator must have a way to detect collisions and an efficient procedure for splitting the workspace into regions containing and not containing objects. To make the search more efficient, subprocedures for maintaining

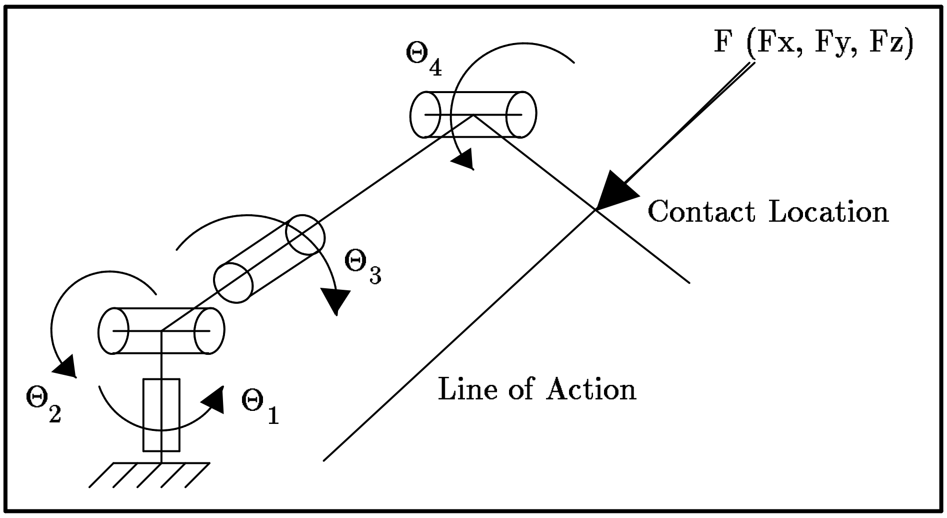

Figure 2.6: **Identifying a contact.** The four degree-of-freedom manipulator can determine the contact location, line of action, and the magnitude of the force in this contact example.

contact with an object should be defined. In addition to the geometry of the workspace, the manipulator can be used to identify the apparent stiffness of the objects and the frictional properties of the objects. Contact detection can be done with tactile sensors, proximity sensors, or by sensing the joint torques. We choose to explore this last method since it extends the utility of the robot by using it directly as a sensor.

Consider the four degree-of-freedom robot shown in figure 2.6. This manipulator can sense four joint torques along the axes labeled θ_1, θ_2, θ_3, and θ_4. By assuming that the contact force is a single contact along the last link with no torque about the contact, the contact location and the force vector can be determined in most configurations. The contact force contains three parameters (F_x, F_y, F_z) and the contact location is a single parameter (L, the distance along the link where the contact occurs) for four

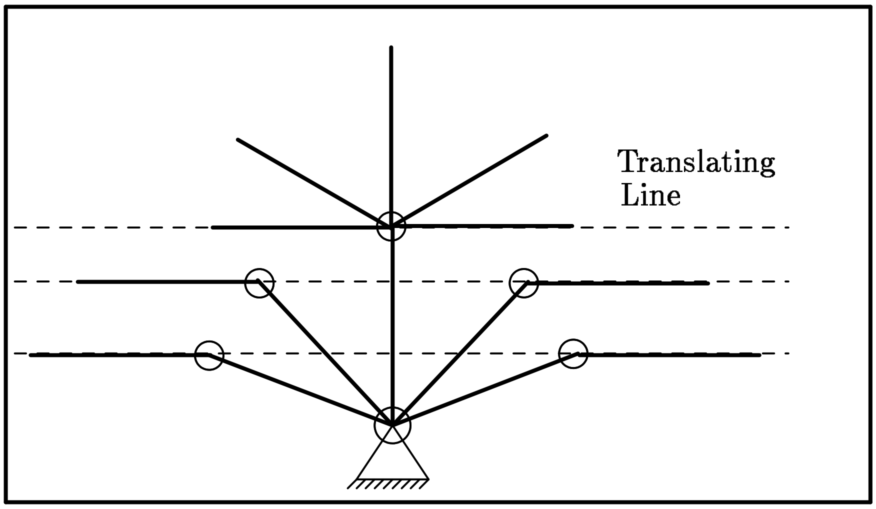

Figure 2.7: **Searching a plane with line motions**. The figure shows a two degree-of-freedom manipulator searching a plane with line motions. The line translates in one direction, and then reverses direction when the manipulator reaches the end of its workspace. By using a line, the amount of area searched is much greater then with an endtip point search.

measurements and four unknowns. For a given L, the torques are equal to the product of the forces with the manipulator Jacobian transpose. However, in the contact problem the Jacobian is a function of the unknown contact location L, implying that a closed form solution is not equivalent to the inverse of the Jacobian transpose. In section 4.2.3 a numerical solution to the contact problem is derived and issues relating to convergence and error conditioning are discussed.

A simple partition of the workspace can be created by moving the links along surfaces of interest until a contact is obtained. The surfaces could be specified by the motion of two points on the surface as a function of time. The line connecting the two points would then specify the motion of the manipulator. By using lines, the amount of area which

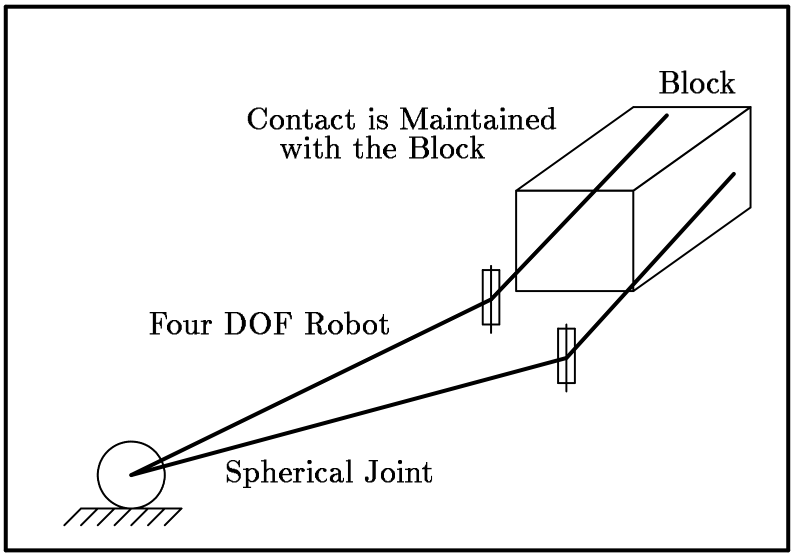

Figure 2.8: **Mapping the Shape of a Block.** After making contact with the block, the manipulator maintains contact with the block as it continues the search.

is eliminated from consideration is significantly greater then for an endtip search (figure 2.7) but, the resolution is less.

For the contact portion of the search, the manipulator conforms to the object and performs a compliant search. For example, in the case of the block shown in figure 2.8, the manipulator would apply a small bias force into the block to maintain contact as it moved over the block. Additional information can be gleaned from the search by rocking the end-effector and noting the maximum distance along which the contact can be maintained. More sophisticated techniques for search and object identification, or haptics, are an active area of research.

2.2.3 Enclosure

Enclosure is defined here as the creation of a kinematic structure around an object such that the object is at rest in a stable equilibrium. Grasping and enclosure are effectively equivalent, where we extend grasping to include grasps using robot links. While there have been many analytic results for fingertip (dexterous) grasps, few analytic results have been derived for innerlink (power) grasps.

With a whole-arm-manipulator, we are presented with a manipulator, or a set of manipulators, that can perform both types of grasps. Figure 2.9 shows a set of WAM manipulators performing an inner link grasp on a set of cylinders. Figure 2.10 shows a set of manipulators performing a dexterous grasp on a block. This concept of WAM manipulators as large hands can be extended smaller, and larger, by having small manipulators attached to the large manipulator at convenient locations.

There are two ways to perform an enclosure with the inner links. In figure 2.9, a rigid kinematic structure is formed by the manipulators' links and then the objects are placed into this structure. In this case, the objects settle under the force of gravity into the constraining shape defined by the manipulators' links. Alternatively for a power grasp, a compliant kinematic structure, formed from the manipulator's links, can be placed around the object, and then the stiffness of the structure can be increased while keeping the structure in contact with the objects. In this type of grasp, the manipulator conforms to the object. For both types of grasps there are a large number of possible contact points and geometries which will form a stable grasp. Therefore to generate an enclosure, we suggest that the final contact location need not be specified; rather, a technique for

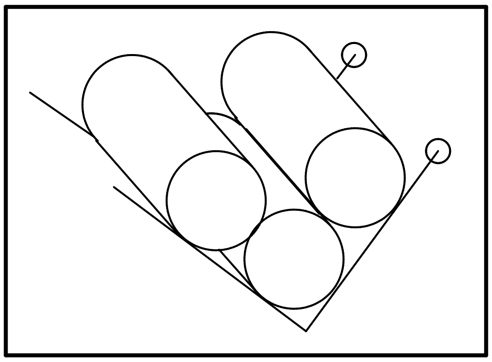

Figure 2.9: **Holding Firewood**. Two WAM manipulators are used in this example to perform an inner link grasp on a set of cylinderical objects.

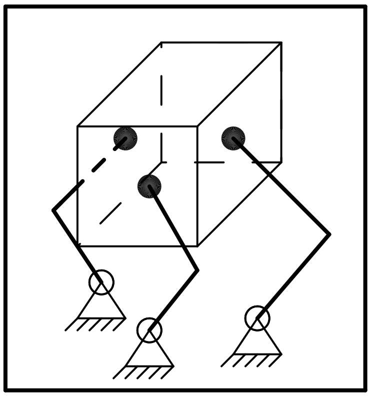

Figure 2.10: **Holding A Block**. Three WAM manipulators are used in this example to perform a dexterous grasp on a block.

generating a set of line motions such that the object is guaranteed to lie within a stable contact region should be specified. This type of procedure would be inherently robust to errors in robot calibration and location measurements. Such a procedure, is a topic for future research.

2.2.4 Exclusion

Exclusion is defined here as the creation of a kinematic structure that tends to prevent objects from entering a defined region. Although this structure will push on objects which attempt to enter the region, exclusion is different from pushing because the robot is not trying to move any particular object at a given instant. The structure must be stable under impulse loadings and the holes in the structure must be sufficiently small. The size of the holes must be smaller then the smallest projection of any object of interest onto a plane. Exclusion includes catching and blocking objects.

2.3 Summary

All of the basic tasks, pushing, searching, enclosure, and exclusion, involve contact with the environment along link edges. The kinematics of each task can be specified by the motion and position of lines through the links. In each example, the contact location along a link was not specified; instead, a line of possible contact locations and the direction of link motion was specified. Because of the large number of possible contact locations, the procedures are robust to errors in locating the object. By allowing contact along the whole manipulator, the range of object sizes and weights which can be

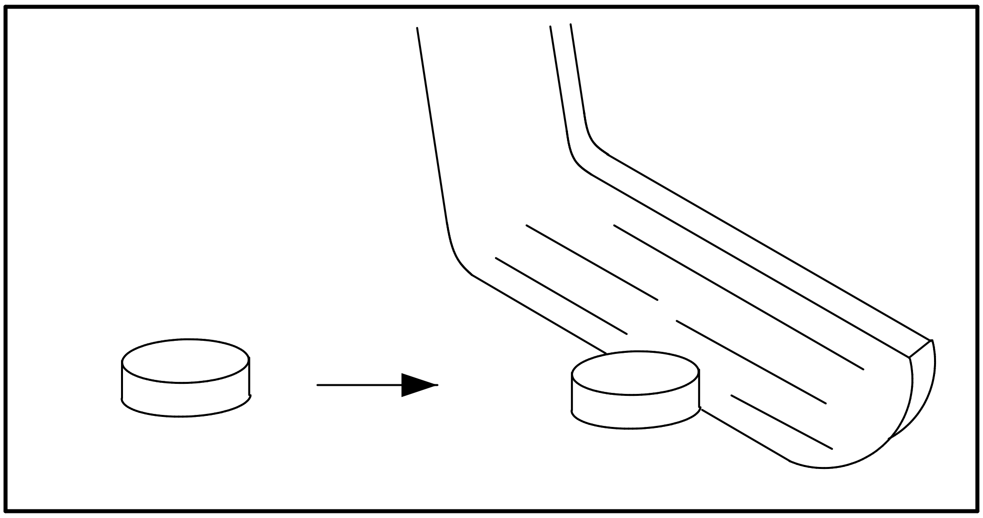

Figure 2.11: **Hockey Stick Catching a Puck.** An Example of Exclusion

manipulated is extended beyond what would be available with end-effector grasping.

Control for the above set of tasks requires the ability to apply and control the interaction forces while moving across the workpiece. This requires an understanding of the control issues involved with force control and is discussed in chapter 3.

Chapter 3

WAM Control

This chapter addresses the issues involved with controlling a whole-arm manipulator so that it interacts with the environment in a stable and useful manner. We will review some of the approaches that have been taken by others to solve the problem of force control or compliant control. We will, then, discuss some recent results in the literature which are the motivation for our approach to whole-arm control.

3.1 Requirements

Whole-arm manipulation involves doing work on objects in a controllable and stable manner. The control system must be robust to a wide range of environment impedances, contact locations, and manipulator kinematic configurations. The manipulator must be able to collide with objects without causing excessive forces. During contact, the system must be able to apply and measure the forces of interaction.

A whole-arm manipulator is designed to interact with the environment along all of its moving surfaces. Since the effective impedance of the environment, for a point contact, varies with the square of the contact's distance from the joint, the contact impedance varies widely as the contact location changes. The impedance for a line contact can change dramatically with a small change in joint angles. A line contact has the interaction force distributed along the length of the link; consequently, materials usually considered soft will appear to be quite stiff at the manipulator joints. A slight change of angle could change a line contact along a box to an edge, or point, contact causing a large change in the impedance.

The initial contact between the manipulator and the environment can cause sudden impact loads, to which the system must be robust. In addition to stability, we require that the controller exhibit smooth transitions from free motion to constrained motion without excessive bouncing. The forces induced by impact must be minimized not only to prevent both damage to the manipulator and the object, but also to prevent moving the object during a searching procedure. The implication is that the speed of approach of the manipulator must be low and that the manipulator and controller must be designed to minimize the initial contact loads.

Lastly, the manipulator must be able to apply a desired force and measure the forces of interaction. The forces could be controlled directly by closing a force control loop with sensors along the manipulator, or the forces of interaction can be inferred from the commanded/sensed joint torques. The reasons we have chosen to use the joint torques to control the forces of interaction are presented in the next section. This choice has implications for the design of both the controller and the robot mechanism.

3.2 Compliant Control/Force Control

The control of the forces of interaction between a manipulator and the environment has been the focus of considerable research under the general heading of force control. There are many implementations of force controllers. Approaches which make use only of the interaction forces are termed *explicit force controllers* [Nevins and Whitney 73]. Other methods use both force information and position or velocity information to control force. These methods include *stiffness control* [Salisbury 80], *damping control* [Whitney 77], and *impedance control* [Hogan 84]. All of these methods use force information to alter position or velocity loops. *Hybrid control* [Raibert and Craig 81], resolves the robot path into directions of either pure force control or pure position control. All of these schemes are reviewed in [Whitney 85]. The approach in this thesis is based on stiffness control which can be considered a subset of impedance control.

Many experimenters have found that force control algorithms are difficult to stabilize and that limit cycles within the system or, chattering on the surface of the workpiece, often result. In order to stabilize the system, the bandwidth of the force controller has often had to be reduced. Eppinger [Eppinger 87] examined the role of manipulator structural dynamics and compensator dynamics on instability and bandwidth. Lawrence [Lawrence 88] examined the effects of computational time delays on the bandwidth of an impedance controller. An [An 86] viewed force feedback as high gain position feedback and found that unmodeled high frequency dynamics caused stability problems. Wlassich [Wlassich 86] discovered that if the desired mass for an impedance controller was less than the actual mass, the manipulator became unstable against a stiff environment. The

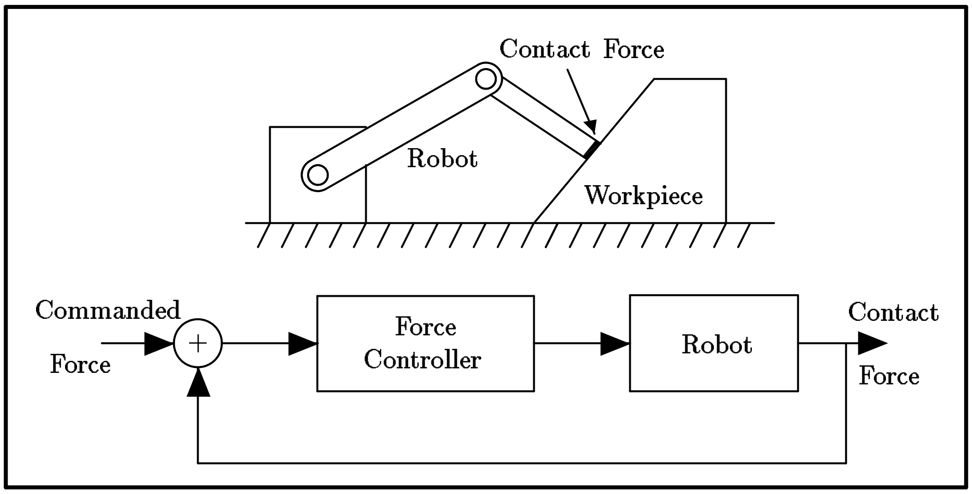

Figure 3.1: **A Simple Robot Compliance Control Task**

effects of force feedback and the difficulty of force feedback stabilization can best be understood in the framework presented in [Hogan 84].

3.2.1 Impedance Control

A requirement of whole-arm manipulation is the ability to do work on the environment. This implies that along any direction in which work will be performed both velocity and

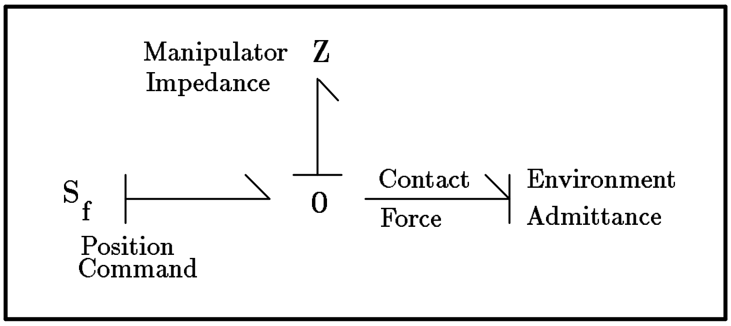

Figure 3.2: **Bond Graph Equivalent Network for Robot Compliance Control**

forces will be generated, and the controller needs to assume some level of control over both. Causality arguments show that it is impossible for a system to dictate both the velocity and the force at a given contact [Rosenberg and Karnopp 83], but that the relationship between the contact force and the contact velocity (the manipulator impedance) and the contact velocity may be specified. This approach to contact control is known as impedance control.

The robot's environment may appear as either an admittance, which accepts forces and yields motions or as an impedance which accepts motions and yields forces. In practice most of the objects in the environment appear as fairly rigid masses and are thus best modelled as an admittance. For example, a block sliding on a table presents an inertial mass and a frictional resistance to an applied force. The mass accepts force inputs and integrates the force to give the position and velocity of the mass at any given time. It does not make physical sense to talk about the mass accepting a velocity input since this would require infinite force. A consequence of the environment acting as an admittance is that the manipulator should appear as an impedance. Hence, the manipulator should accept motion commands and return forces.

An impedance control algorithm is specified by the desired effective endtip mass, stiffness, and damping at the manipulation location. A controller is implemented which causes the manipulator to approach this behavior at low frequencies. The desired impedance is given as:

$$[Is^2 + Cs + K]\Delta X(s) = \Delta F(s), \tag{3.1}$$

where s is the Laplace operator, I is the desired inertia matrix, C is the desired damping,

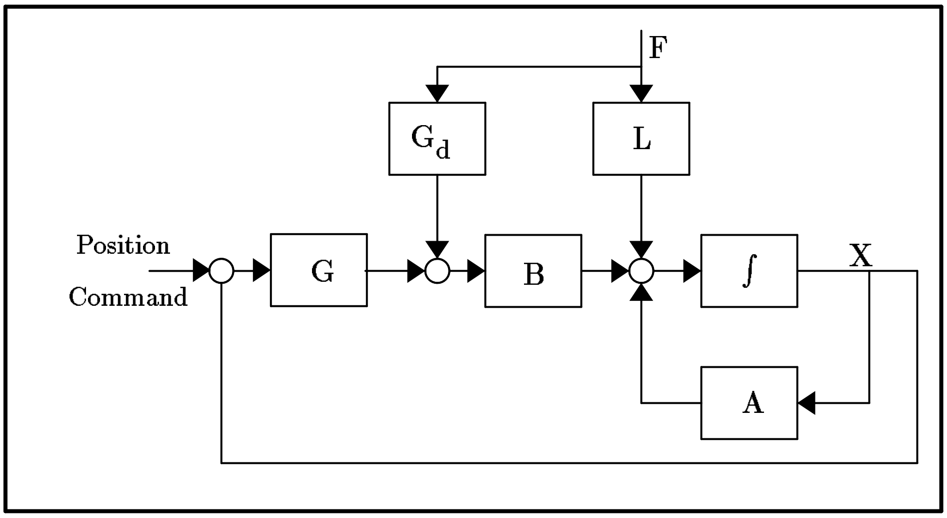

Figure 3.3: **Closed Loop Dynamic System: Robot interacting with the environment.** F is the interaction force, X is the vector of plant states, G_d is the force feedback gain, L is the transfer function from the environmental forces to joint torques, $F(x, u, t)$ is the robot plant dynamics, and G is the full state feedback transfer function.

K is the desired stiffness, X is a vector of generalized displacements, and F is the vector of generalized forces. The problem is to find gains G_d and G (shown in the general control structure in figure 3.3) to make the manipulator approach the target dynamics for low frequencies. A nonlinear impedance control algorithm which accomplished this can be found in [Wlassich 86]. An eigenstructure assignment approach for the small motion linearized dynamics can be found in [Kazerooni 85].

The dynamics of the robot can be expressed in nonlinear form as:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau, \tag{3.2}$$

where H is the inertia of the robot, C is the matrix of centrifugal and coriolis effects, G is the vector of gravity induced torques, and q is the vector of generalized joint positions

[Asada and Slotine 86]. This equation can be linearized about an operating point to produce the equation of motion:

$$J\Delta\ddot{X} + B\Delta\dot{X} + S\Delta X = \Delta F. \tag{3.3}$$

Kazerooni shows that if the desired mass I is equal to the actual manipulator mass H, expressed in the coordinate system of the impedance controller, no force feedback is necessary.

Similarly, Wlassich found that the effect of force feedback in an impedance controller was to modify the apparent mass at the end-effector. Negative force feedback decreases the apparent mass and positive force feedback increases the apparent mass. Without force feedback, the desired mass must equal the actual manipulator mass.

Impedance control can be implemented in two ways. An inner, stiff position and velocity loop can be implemented on the robot and an outer force loop can be used to decrease the effective stiffness. Alternatively, an inner loop based on torque, and an outer position or velocity loop can be implemented.

The first method, position based impedance control, is the approach most often studied. Most industrial robots come with very stiff inner position/velocity controllers because this type of controller masks some of the inherent mechanical nonlinearities in the transmission and actuator. The effects of torque ripple, motor brush friction, gearing friction and backlash are all reduced by the addition of a high gain velocity controller. However, the endpoint force controller then becomes a non-colocated controller and the intervening actuator and robot dynamics limit the gain that can be achieved ([Cannon 83], [Eppinger 87], [Kazerooni 85], and [An 86]).

Figure 3.4: **Effect of Time Delay on the Stability Boundary for Position Based Impedance Control.** Results are representative of an experiment by [Lawrence 88].

Figure 3.5: **Effect of Time Delay on the Stability Boundary for Torque Based Impedance Control.** Results are representative of an experiment by [Lawrence 88].

The effect of the computational time delay has a very different effect for the torque based versus the position based impedance controller. Figure 3.2.1 shows the effect of the time delay on the two different implementations. For a position based system, low stiffness can only be achieved by making the target impedance highly overdamped but, high stiffness can be achieved with little or no target impedance damping. The torque based approach has the opposite characteristic. High stiffness can only be achieved with significant damping in order to cause the target impedance to be critically damped. Similarly low stiffness require low damping.

For a whole-arm manipulation controller, we require critically damped response for a wide range of stiffness values. This coupled with the difficulties presented by the unmodeled dynamics in the position based approach lead us to the development of a system based on torque based impedance control and no interaction force feedback.

3.3 Torque Based Impedance Control

In a torque based approach to impedance control, an inner loop based on torque is placed around each joint actuator. This inner loop can be based on the current into the motor, with a linear relationship assumed between the motor current and the motor torque, or joint torque sensors can be placed on each motor. A few researchers have used joint torques to control the interface forces between the robot and the environment. Wu and Paul [Wu 80] implemented a joint torque sensor for a single joint manipulator. They used an inner velocity loop on the actuator for damping and closed the torque loop around a harmonic drive transmission. Luh, Fisher, and Paul [Luh 83] implemented torque servos on each of the joints of the Stanford manipulator. The torque servos were used primarily to reduce the frictional effect in the joints and harmonic drive. They observed that a limit cycle occurred in the control loop due to the backlash in the harmonic drive. An [An 86] used the motor currents to infer the joint torques on the MIT-DDARM. He experimented with using only the motor currents, the motor current in conjunction with a force sensor, and using a force sensor alone.

Because joint torque control systems are colocated controllers, they have inherently higher bandwidth. Further, for an impedance controller, the stiffness of the controller is directly controlled by the position feedback gain, and the range of stiffnesses is not as limited by the computational delay. However, this type of implementation has some inherent drawbacks. Since the controller is colocated, the acceleration of the effective manipulator mass effects the applied force. Without force sensing, we lose the ability to modulate the apparent mass through feedback. Any nonlinearities in the motor and transmission which occur after the torque feedback will not be attenuated by the force control loop. Any friction, motor ripple, stiction, and backlash will be felt at the open-loop values. Therefore, the manipulator must be designed to minimize these effects. Because the target mass is identical to the desired mass, a torque based impedance controller reduces to a stiffness control algorithm. The control law becomes:

$$\tau = G(q) + J^T K J \tilde{q} + B\dot{\tilde{q}} \tag{3.4}$$

$$\tilde{q} = q_d - q, \tag{3.5}$$

where $G(q)$ is the vector of gravity compensation terms, J is the Jacobian matrix for relating the joint velocities to the generalized velocities at the contact location, K is a symmetric positive definite matrix of stiffness gains, B is a symmetric positive definite matrix of damping gains, q is a vector of generalized joint positions, and q_d is a vector of desired joint positions. The algorithm can be shown to be stable in all configurations (see [Asada and Slotine 86]).

3.3.1 Mechanical Issues and Effects

There are two approaches to torque based force control. The torque can be sensed, or the motor current can be sensed and the torque inferred from the motor current. We chose to use the motor currents which reduces the number of sensors required in the system and increase the reliability of the overall system. However, we must face the problems of torque ripple, cogging, and deadband in the motors; friction, backlash and

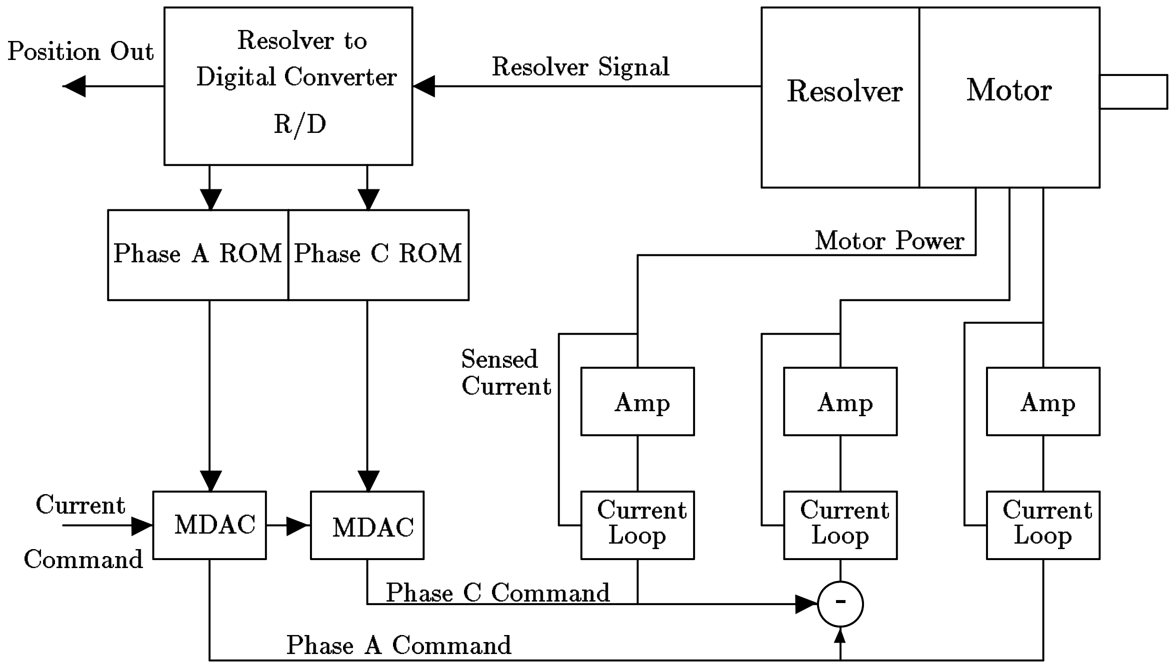

Figure 3.6: **Configuration of a Brushless Motor Controller**

deadzone in the transmission, and large bias forces from the gravity loading. Each of these effects must be modelled and compensated. Each of these effects is examined in the next sections; and compensation and design approaches to minimize the effects are suggested.

Brushless Motors: Torque Ripple, Cogging, and Deadzone

A typical three-phase brushless motor and controller has a configuration similar to the one shown in figure 3.6. The controller gets position feedback through a brushless resolver. The position data is used to select commutation values for two of the three phases. Each of the computed values is then multiplied by a current command. At this point the third phase is generated by subtracting the first two phases. A current loop is then closed on each of the phase's sensed current, using either hall effect sensors or resistors, to determine the motor voltage necessary for each phase.

The advantage of the brushless motor is the lack of brush friction; however, brushless motors can have as much or more ripple and cogging as brushed torquers because presently available motors have fewer poles. Torque ripple is the variation in motor torque with angular position and is quoted as a percentage of commanded torque. Cogging, is the variation in motor torque that occurs when no torque is being commanded, and is usually a fixed value independent of the commanded torque. Lastly, most motor controllers have a torque deadband.

One source for these problems is current loop deadband in the controller. Experimentally Paul [Paul 87] showed that deadband in the current loop produces ripple at all integer multiples of the fundamental driving frequency. Significant ripple and cogging can also arise from the spatial harmonics in the magnetic field within the motor.

The effects of cogging and ripple can be reduced by modifying the commutation tables in the motor controller or by adding a commutation table to the servo computer to modify the commanded torque. The latter approach is used in this thesis. By carefully measuring the zero-speed ripple and cogging, a table of torque correction values was created that reduced the cogging from 1.0 in-lb to 0.2 in-lb for a 13.5 in-lb torque command. Additional information on implementation, measurement and velocity dependence is presented in section 5.2.1. The deadzone in the motor controller can be compensated as though it were joint friction and is examined in the section on friction compensation.

Backlash and Keys

Backlash can be present in any mechanical system that uses gears to transmit power or keyways to couple shafts. Backlash in the system degrades performance in a number of ways. Inside a control loop, it can cause limit cycles damaging to the components, and position errors in motor based position measurements.

In a previous implementation, [Salisbury 88], a keyway coupling a single joint test bed to the drive motor was found to decouple the joint from the motor for small motions. This led to a decrease in effective damping for the joint and to a nonlinear behavior in the overall system. This behavior was manifest by excitation of the first dynamic mode of the system, at 60 Hz, by a command to the motor at 20 Hz.

In any implementation of a whole-arm manipulation system, it is very important that backlash between the actuators and the output links be eliminated. This is accomplished in the MIT-WAM manipulator, by the use of terminated cables as a transmission mechanism. This transmission has been shown to have zero backlash and very low friction. A detailed description of the transmission and the mechanical design principles for the transmission can be found in [Townsend 88] and [DiPietro 88]. The transmission is connected to the motor through a tapered shaft which eliminates the backlash of the keyway.

Joint Friction and Controller Deadzone

Friction in the system takes a form similar to Coulomb friction in the joints, viscous losses in the motor bearings, and in the command deadzone in the motor controller. We have found that with a cable transmission, the greatest source of friction becomes the motor.

In order to compensate for the Coulomb friction, in the context of our stiffness controller, we have designed a switching compensator which applies an additional torque in the direction of desired motion. This control law takes the form of:

$$\tau_f = -\hat{\mu}\, sgn\,(\tilde{q}), \tag{3.6}$$

where τ_f is small correction torque for friction, $\hat{\mu}$ is an estimate of the magnitude of friction, sgn is the sign function, and $\tilde{q}$ is the difference between desired and actual joint coordinates defined in equation 3.5.

This control law has the effect of compensating for friction when the system is moving toward the desired location and adding additional friction when the system is moving away from the desired location. Thus, the net effect is to drive the system toward the goal location and to overcome friction. An analysis of a single degree-of-freedom system with no viscous damping demonstrates the effect.

Consider a dynamic system with the equation of motion:

$$\ddot{x} + x + \mu sgn(\dot{x}) = F. \tag{3.7}$$

This equation describes a spring and a mass with Coulomb friction driven by a force F. Now consider compensation of the form of equation 3.6. The combined system in phase space is:

$$\dot{x} = y \tag{3.8}$$

$$\dot{y} = -x - \hat{\mu}\, sgn\,(x) - \mu\, sgn\,(y). \tag{3.9}$$

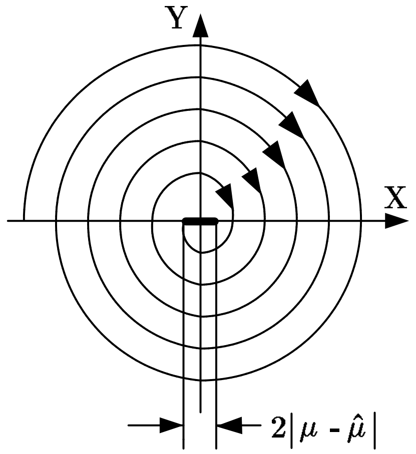

Figure 3.7: **Phase Plane Plot of** $\ddot{x} = -x - \hat{\mu}\, sgn\,(x) - \mu\, sgn\,(\dot{x})$ **for** μ **small.** The range of critical points is $2|\mu - \hat{\mu}|$.

If $\mu = \hat{\mu}$, the friction term and the compensation term will cancel in the II and IV quadrant of the phase plane (i.e. the system will move without apparent friction.) The only critical point for this equation is at the origin. In the I and III quadrants, the system will appear to have twice the friction. A locus of critical points exists for $y = 0$ and $|x| \leq 2|\mu - \hat{\mu}|$ for this equation. The intersection of these critical points gives only the origin as a true critical point and thus the system spirals in, as in figure 3.7.

A more rigorous proof can be given by defining the Lyapunov function:

$$V = 1/2x^2 + 1/2\dot{x}^2. \tag{3.10}$$

This function is always positive except when both the position and velocity of the system are zero. Taking the derivative of V:

$$\begin{aligned} \dot{V} &= \dot{x}x + \dot{x}\ddot{x} && (3.11) \\ &= -\dot{x}(\hat{\mu}\, sgn\,(x) + \mu\, sgn\,(x)). && (3.12) \end{aligned}$$

For $\hat{\mu} \leq \mu$, examination of all of the possible combinations of $sgn\,(x)$ and $sgn\,(\dot{x})$ shows that $\dot{V}$ is zero only when $\dot{x} = 0$ and $x = 0$, or in the case when $\hat{\mu} = \mu$ and $\dot{x}$ and x have opposite sign. In the latter case the system equation becomes $\ddot{x} + x = 0$ which is stable, and will eventually enter a quadrant where $\dot{V}$ is less than zero. Therefore, with an an estimate of friction less than or equal to the actual value of friction, this compensator is stable and causes the system to approach the desired location with greater precision.

Additional analysis can be performed for small values of μ, using the method of averaging, to show that, for small values of $\hat{\mu}$ and μ, $\hat{\mu}$ can be greater than μ and stability will still hold. The analysis also shows that friction, not the compensation, removes energy from the system, and that the compensation has the effect of changing the phase of the initial condition time response. This analysis is detailed in Appendix A.

Gravity Loading

The loads on the manipulator induced by gravity are also of significance for the torque based approach. The gravity loads produce a large position dependent bias force in the torques; which must be subtracted from the joint torques to calculate the torque due to contact.

The affects of gravity can be reduced by effective robot design. By placing the actuators on sections of the robot that are supported and transmitting the power through a light weight transmission to the joints, the effective mass of the manipulator and the torques due to gravity can be significantly reduced. This was done in the MIT-WAM manipulator through the use of a cable transmission to actuate the last joint (see the drawing of the mechanism in figure 4.1).

Atkenson [Atkenson 86] shows that the torques due to gravity are linear in the parameters that determine the loading. Thus the gravity loading is:

$$G(q) = F(q)\hat{a}, \tag{3.13}$$

where $F(q)$ is a function of the kinematics and $\hat{a}$ is a set of parameters that determine the loading for a particular geometry. The parameters can, therefore, be determined by measuring the torques due to gravity in a number of locations and performing a linear regression on the data.

Arm Mass and Transmission Design

In addition to the gravity loading, the mass distribution of the robot effects the inertia loads and the contact forces. For torque based force control, the inertia of the arm should be minimized in order to match closely the torques from the motors with the forces on the environment. Because our current control implementation lacks a model of the inertial and Coriolis accelerations, we have found it necessary to limit the desired arm acceleration.

By equating kinetic energy before collision to potential energy during collision, the maximum impact force, F_i, is approximated by:

$$F_i = v_i\sqrt{J_l k_c},$$

where J_l is the inertia of the arm along the line measured to the point of contact, k_c is the contact stiffness, and v_l is the contact velocity. In order to minimize the impact forces, the velocity of approach should be limited, the robot should be designed to minimize the backdriven mass, and the contact compliance should be increased. Thus, the acceleration-dependent backdrivability should be maximized for a whole-arm manipulator. More details on the requirements for good backdrivability can be found in [Townsend 88].

3.3.2 Line Stiffness

The stiffness term, K of $J^T K J$, in the feedback law can be used to control the manipulator in different spaces. As developed in chapter 2, line motions are one possible representation for whole-arm tasks. Thus, we introduce the concept of a line stiffness.

A line stiffness is a 4x4 stiffness matrix relating the restoring forces and moments to the incremental motions of a line. In order to describe the differential motion, we affix a frame to the line at any desired location. The orientation of the frame about the line is also chosen. For a line, motions along the length of the line and rotations about the line axis are not defined. The remaining four possible motions are related to the restoring forces and moments by the stiffness matrix.

In a whole-arm task with stiffness, the controller updates the Jacobian and the stiffness matrix as the task proceeds. For a search task, with a line edge contact, the location of the line frame could be defined as the contact location. To prevent large forces in the contact direction, the line stiffness in the direction of contact should be chosen small.

As the contact location moves along the length of the manipulator, the line frame would move with the contact, since the Jacobian relates joint velocities to line velocities at the contact point, motion of the contact causes an update of the Jacobian. This update keeps the center of stiffness at the contact location and the stiffness in the contact normal direction small. Line stiffness are, therefore, applied to a whole-arm task by defining the required stiffnesses at a line frame and updating the location and orientation of the frame.

3.3.3 Summary of Control Approach

In summary, the control algorithm that is used for our implementation of whole-arm control is based on an impedance control formulation with the desired mass equal to the actual manipulator mass. This choice reduces the impedance controller to a stiffness controller.

The inner-loop of the controller is based on torque, giving the controller high bandwidth, and good stability margins. The choice of an inner-torque loop without an outer force loop implies that the apparent manipulator mass cannot be modified through feedback control. The manipulator, therefore must be designed to have a low effective endtip mass or high acceleration-dependent backdrivability.

In addition, the lack of a distal force sensor implies that friction and backlash in the joints and transmission cannot be attenuated through feedback control, although, a feedforward controller can reduce the effects of friction. The design of the manipulator must inherently minimize these effects.

Lastly, because of the lack of a torque sensor in the inner-torque loop, the nonlinearities in the motor cannot be attenuated by feedback. The motor's torque ripple and cogging can be attenuated only through feed-forward compensation.

The final control algorithm contains position and velocity feedback to implement the stiffness controller, a gravity feed-forward controller, a switching controller to attenuate friction, and a torque ripple feedforward term to attenuate ripple and cogging. The terms are summarized as:

$$\tau_{stiffness} = J^T K J \tilde{q} + B\dot{\tilde{q}} \tag{3.14}$$

$$\tau_{gravity} = F(q)\hat{a} \tag{3.15}$$

$$\tau_{friction} = F_{friction}\, sgn(\tilde{q}) \tag{3.16}$$

$$\tau_{ripple} = \Delta\tau_{ripple}(q), \tag{3.17}$$

where the values of $\hat{a}$, $F_{friction}$, and $\Delta\tau_{ripple}$ must be experimentally determined, and $F(q)$ depends on the manipulator kinematics. The implementation of this controller will be discussed in chapter 5.

3.4 Conclusion

A whole-arm manipulation system can interact with the environment with all of its moving surfaces. Interaction involves issues of trajectory planning and compliant control. The previous chapter provided a possible taxonomy of tasks for whole-arm manipulation and suggested that each of these tasks could be represented by the motion of lines in space. Effective control of the lines can be achieved by the use of line compliance.

In this chapter, we have examined the issues involved with force control of manipulators. A whole-arm task does work on the environment; therefore, the controller must assume some level of control of both the velocity and force along every axis. To control the relationship between force and velocity, we have implemented a stiffness controller with feedforward compensation.

The robustness of the system can be increased by implementing the controller as an inner joint torque controller and an outer position controller. This implementation is an example of colocated control and does not suffer the limitations on bandwidth imposed by computational time delay and manipulator dynamics. The robustness of the controller is further enhanced by the use of motor currents to infer joint torques.

However, by eliminating a interface force sensor the nonlinear effects of the transmission and motor controller cannot be mitigated by feedback control. Effective mechanical design and feedforward compensation techniques must be applied.

The remaining chapters are a case study of the MIT-WAM manipulator for the concepts outlined in chapters 2 and 3.

Chapter 4

Manipulator Kinematics

The kinematics of a manipulator with n degrees of freedom is generally described as a mapping from the n generalized joint coordinates to the location and orientation of the endtip frame in a global cartesian frame of reference. This function is termed the forward kinematics. Texts such as [Craig 87], and [Asada and Slotine 86] describe procedures for solving the forward problem.

The inverse problem of taking the desired endtip location and solving for the required joint angles is considerably more difficult and results in a one-to-many mapping if any solutions exist. For manipulators with more freedoms than constraints, the mapping is under-specified and additional constraints must be imposed.

Additional constraints can be derived from obstacle avoidance requirements, workspace optimization, energy or torque minimization, or other criteria. Note that the need to specify additional internal constraints arises because the task is being specified in terms of locating the end-effector of a robot with n freedoms (with $n > 6$ for a spatial manipulator) in cartesian space which can provide at most six constraints. If other kinematic constraints were employed, the kinematic mapping would not be under-specified. Obstacle avoidance, bracing, pushing, searching, enclosure, and exclusion are all instances of motions with additional kinematic constraints. In general these mappings specify where intermediate links of the robot must or must not lie. For the MIT-WAM, the specification of the location of a line on which the last link must lie provides four constraints which are sufficient to specify a kinematic mapping.

In this chapter, we explore the forward kinematics and inverse kinematics for the MIT-WAM. After a description of the manipulator geometry, we derive the endpoint inverse kinematic relationships. This set of relations is derived both to move the endpoint in cartesian space and to understand the contact problem for whole-arm manipulation. The redundancy of the manipulator for endpoint kinematics will be resolved simply by using the θ_3 joint angle as a free parameter which must be specified.

Line kinematics are introduced by a section on the specification of lines and the important properties of lines. This set of properties is applied to the forward kinematic problem of describing the location of the line through the last link of the manipulator given the manipulator joint angles. Then, the inverse kinematic problem of finding a set of joint angles to place the last link along the axis of a given line is derived by a series of geometric transformations. Lastly, the concept of a line Jacobian is introduced.

4.1 Manipulator Geometry

Figure 4.1 shows the locations of the joint axes for the manipulator, which are labeled θ_1, θ_2, θ_3, and θ_4. The θ_1 axis is vertical and has a range of 350°. The θ_2 axis intersects

Figure 4.1: **The MIT-WAM Manipulator**

the θ_1 axis at right angles and has a range of 270°. The θ_3 axis intersects the θ_2 axis and the θ_1 axis at right angles and has a range of 300°. The θ_4 axis is located away from the base and is perpendicular to and slightly offset from the θ_3 axis and has a range of 225°.

Each joint is driven by a cable transmission powered by Moog 300-003 DC brushless motor. (See [Townsend 88] for more details on the transmission.) Each transmission stage includes a two-stage 30:1 reduction (20:1 in the fourth joint) located at the driven joint. The θ_1 transmission drives its joint directly. The θ_2 and θ_3 joints are powered by two motors acting through a cable differential.[1] The last joint is powered from a motor attached at the differential by a long cable transmission passing through the center of link three. The transformation from motor coordinates to joint coordinates is given by:

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{pmatrix} = \begin{pmatrix} \frac{1}{\mathrm{N}_1} & 0 & 0 & 0 \\ 0 & \frac{-1}{2\mathrm{N}_2} & \frac{1}{2\mathrm{N}_3} & 0 \\ 0 & \frac{-1}{\mathrm{N}_2} & \frac{-1}{\mathrm{N}_3} & 0 \\ 0 & 0 & 0 & \frac{1}{\mathrm{N}_4} \end{pmatrix} \begin{pmatrix} \theta_{m_1} \\ \theta_{m_2} \\ \theta_{m_3} \\ \theta_{m_4} \end{pmatrix}, \tag{4.1}$$

where θ_{m_i} are the motor positions and

$(\, \mathrm{N}_1 \quad \mathrm{N}_2 \quad \mathrm{N}_3 \quad \mathrm{N}_4 \,) = (\, 29.97 \quad 30.40 \quad 30.40 \quad 19.15 \,)$.

4.2 Endtip Kinematics

The endtip cartesian kinematics is derived in this section. Homogeneous transform notation will be used to derive the forward kinematics. This convention is widely used, though with slightly different link numbering schemes. We will use the approach detailed in [Craig 87]. The forward kinematic transforms can then be used to propagate

[1] MIT has applied for patent protection of the concept on which the design of this cabled differential is based.

joint velocities out to the last frame to find the differential or "Jacobian" relationship between the endtip velocity and the joint angular velocities in the last frame. This relationship is applied to the collision detection and location problem described in section 2.2.2. Lastly, the inverse kinematic relationship for the endpoint location is derived. This relationship has a single degree of redundancy which is resolved by allowing θ_3 to be a free parameter set by the programmer.

4.2.1 Forward Kinematics

The kinematics of the manipulator is defined by a set of frames shown in figure 4.2. The frames follow the conventions for the Denavit-Hartenberg notation as detailed in [Craig 87].[2]

The Denavit-Hartenberg parameters for the arm, with this set of frames, are listed in the following table:

coordinate frame number i	link twist (radians) α_{i-1}	link length a_{i-1}	link offset d_i	joint angle θ
1	0	0	0	θ_1
2	$\pi/2$	0	0	θ_2
3	$-\pi/2$	0	L_3	θ_3
4	$\pi/2$	A_3	0	θ_4
5	$-\pi/2$	$-A_4$	L_4	0

[2]This convention differs from other authors' conventions in the location of frame i with respect to joint i. Craig places the frame i-1 at the beginning of link i-1.

L_3 and L_4 are the inner and outer link lengths, 22 inches and 17.5 inches. A_3 and A_4 are the offsets in the manipulator, 1.6 inches and 1.1 inches.

- α_i is the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about the $\hat{X}_i$ axis.
- a_i is the distance from $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured along the $\hat{X}_i$ axis.
- d_i is the distance from $\hat{X}_{i-1}$ and $\hat{X}_i$ measured along the $\hat{Z}_i$ axis.
- θ_i is the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about the $\hat{Z}_i$ axis.

The kinematic equations for the manipulator can be derived from these parameters using Craig's conventions. Equation 4.2 gives the general formula for the homogeneous transform ${}^{i-1}_{i}T$ that maps coordinates in frame i to frame $i-1$. This transform can be used to map positions and velocities between frames. The inverse of the transform matrix given by ${}^{i}_{i-1}T$ maps coordinates from frame $i-1$ to frame i. The transforms for each of the frames in figure 4.2 are found by applying equation 4.2 to the parameters in the previous table and are given by ${}^{0}_{1}T$ through ${}^{4}_{5}T$ shown below:

$$ {}^{i-1}_{i}\mathbf{T} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2} $$

$$ {}^{0}_{1}T = \begin{pmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3} $$

$$ {}^{1}_{2}T = \begin{pmatrix} C_2 & -S_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.4} $$

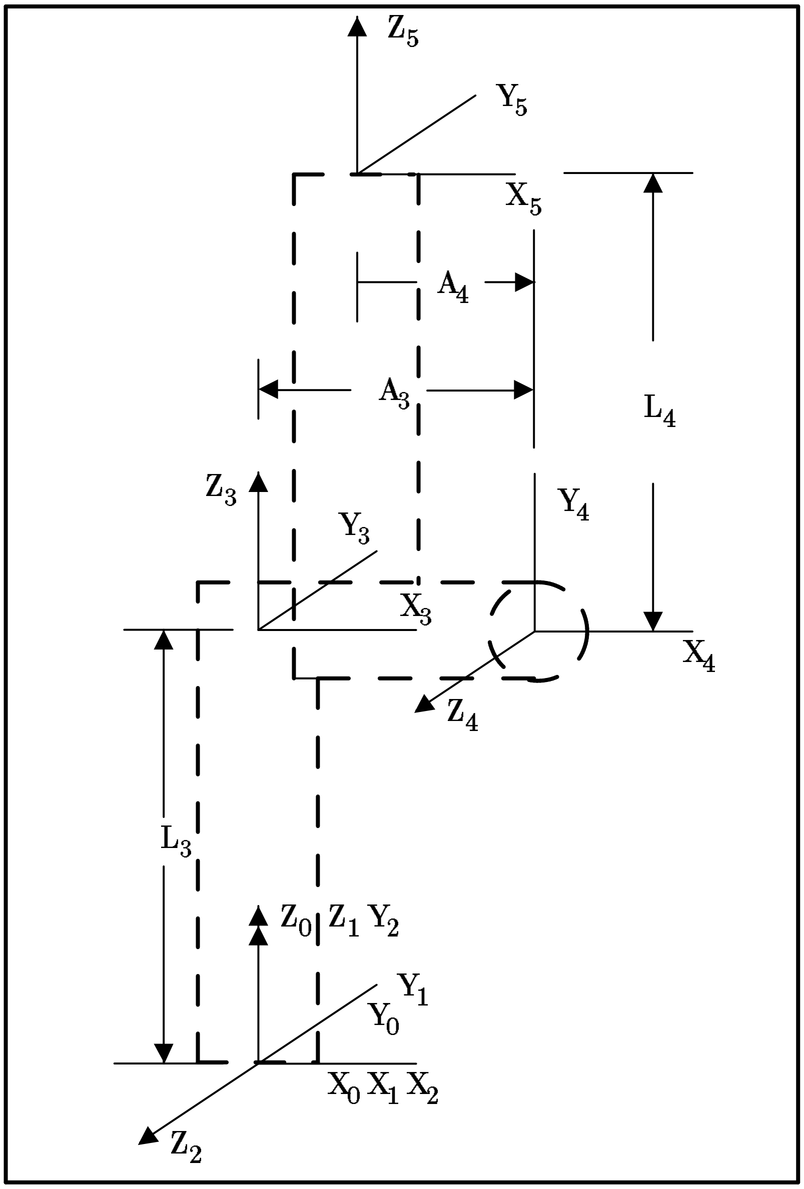

Figure 4.2: **Kinematic Frames for the MIT-WAM**

$$
{}^{2}_{3}T = \begin{pmatrix} C_3 & -S_3 & 0 & 0 \\ 0 & 0 & 1 & L_3 \\ -S_3 & -C_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}
$$

$$
{}^{3}_{4}T = \begin{pmatrix} C_4 & -S_4 & 0 & A_3 \\ 0 & 0 & -1 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}
$$

$$
{}^{4}_{5}T = \begin{pmatrix} 1 & 0 & 0 & -A_4 \\ 0 & 0 & 1 & L_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.7}
$$

C_i and S_i are defined as the cosine and sine of angle i respectively.

For efficiency and convenience each of these transforms has been implemented as two function calls. The first function is the transform function which takes a position vector expressed by $(\,x \quad y \quad z \quad 1\,)$ from frame i to frame $i-1$ by performing the minimum number of calculations. If we express the number of additions by A, the number of multiplications by M, and the number of trigonometric calculations by T, implementation in this way requires $8T + 16M + 25A$ to transform a general position vector defined in frame five to the base frame. Note that there are more addition operations than are obvious from the transform equations because of the need to compute memory address offsets into the position vector; although, address computations are integer operations and the remaining additions are floating point operations.

The second function computes the rotation part of the transform (the upper-left 3×3 matrix). This function is used to compute the orientation of a frame and to transform velocities. Transforming a general velocity vector from frame 5 to frame 0 in this way requires $8T + 16M + 21A$.

4.2.2 Jacobian

If we consider the endtip velocity of our manipulator, a relationship between the endtip linear velocity and angular velocity and and the joint angular velocities can be found by propagating the linear and angular velocities of the inner links out to the last coordinate system by the following iteration:

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_{i} + \dot{\theta}_{i+1}\,{}^{i+1}\hat{z}_{i+1} \qquad (4.8) \\ {}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\left({}^{i}v_{i} +{}^{i}\omega_{i} \times {}^{i}P_{i+1}\right), \qquad (4.9) \end{aligned}$$

where $\dot{\theta}_i$ is the angular velocity of joint i, ${}^{i}\omega_i$ is the angular velocity of link i, and ${}^{i}v_i$ is the linear velocity of joint i. The resulting relationships can be collected into a matrix J that relates the joint velocities in the tool frame (a frame fixed at the manipulator endpoint) to the joint angular velocities as:

$${}^{5}\dot{x} = {}^{5}J\dot{\theta}.$$

The first three rows of this matrix are the relationship between $\dot{\theta}$ and the linear velocities. The last three rows relate $\dot{\theta}$ to the angular velocities. For the cartesian stiffness control algorithm, we will use the first three rows of the matrix. The Jacobian can be transformed into other frames by premultiplying the matrix with the rotation matrix from the tool frame to the desired frame.

The components of the full Jacobian in the tool frame are:

$$\begin{aligned} J[1,1] &= A_3S_2S_3S_4 - (L_4 + C_4L_3)S_2S_3 \qquad (4.10) \\ J[1,2] &= A_3C_3S_4 - C_3L_4 - C_3C_4L_3 \\ J[1,3] &= 0 \end{aligned}$$

$$
\begin{aligned}
J[1,4] &= -L_4 \\
J[2,1] &= (A_4C_3S_2 - C_2L_4)S_4 - (C_3C_4L_4 + C_3L_3)S_2 + \\
C_2(A_3 - A_4C_4) & \\
J[2,2] &= (C_4L_4 + L_3)S_3 - A_4S_3S_4 \\
J[2,3] &= -L_4S_4 - A_4C_4 + A_3 \\
J[2,4] &= 0 \\
J[3,1] &= L_3S_2S_3S_4 + (A_3C_4 - A_4)S_2S_3 \\
J[3,2] &= C_3L_3S_4 + C_3(A_3C_4 - A_4) \\
J[3,3] &= 0 \\
J[3,4] &= -A_4 \\
J[4,1] &= C_2S_4 + C_3C_4S_2 \\
J[4,2] &= -C_4S_3 \\
J[4,3] &= S_4 \\
J[4,4] &= 0 \\
J[5,1] &= -S_2S_3 \\
J[5,2] &= -C_3 \\
J[5,3] &= 0 \\
J[5,4] &= -1 \\
J[6,1] &= C_2C_4 - C_3S_2S_4 \\
J[6,2] &= S_3S_4 \\
J[6,3] &= C_4 \\
J[6,4] &= 0.
\end{aligned}
$$

4.2.3 Contact Determination

In section 2.2.2, it was noted that in certain configurations the contact location and the contact force could be inferred from the joint torques. Salisbury [Salisbury 84a] treats a number of contact problems and demonstrates that it is possible to determine the contact location and the contact force for a number of sensor geometries. In particular for a point contact with friction acting on a hemisphere, six sensors are required to determine the contact force and the contact location. Three parameter are required for each component of the contact force and two parameters are required to specify the contact location.

Similarly for the MIT-WAM five parameters must be determined: three contact forces, the location of the contact along the link, and the angle about the link of the contact. Four joint torques are available to determine the contact force and contact location. By assuming that the contact remains in the same location if the force exerted by the robot is perturbed only slightly, we can take a series of torque measurements to extend the number of sensed parameters and solve for the contact location and contact force. However, this procedure requires additional time and complexity to determine the contact.

Instead we assume that the contact force acts through the center of the last link. This allows an approximate determination of contact location and force components. An estimate for the error created by this assumption can be obtained by determing the torque that is produced by a contact force on the surface of the link about the center of the link. Only the frictional forces produce a moment about the link axis. The magnitude of the frictional force is limited to μ times the normal force. The moment then scales as R, the link radius, times μ times the normal force. The joint torques due to the forces of contact scale as the link length L_3 and the contact location L_4, with the normal force. Therefore for most configurations, the joint torques due to the normal force will be much larger than the torques due to frictional forces. We therefore estimate that neglecting the link radius should produce position estimate errors on the order of μR.

For this thesis then, we consider the contact to occur along an a line through the second link at only one location. Although the torques produced by the offsets are on the same order as the torques produced by the radius of the last link, the offsets are included in the analysis. Because the offsets are constants in the equations relating the forces to the joint torques, the form of the equations is not changed with the addition of the offsets and will not complicate the solution. The cost to adding the offsets is in the additional computation that must be performed in the numerical solution which follows.

Consider again the kinematic frames defined in figure 4.2. In this case let frame 5 be the location of the contact along the last link with the distance L_4 unknown. The joint torques are related to the contact force by the transpose of the first three rows of the Jacobian derived in section 4.2.2.

$$\tau = J^T(L_4)F,$$

where now the Jacobian is a function of the distance L_4. This produces a set of nonlinear equations relating the contact forces and the location of contact to the joint torques. This relationship can be expanded by separating J^T into a part which does not depend on L_4 and a part which does. The joint torques are then related to F and L_4 by:

$$\tau = [A + BL_4]F, \tag{4.11}$$

where A is a function of the joint angles and is equal to the part of J^T that does not depend on L_4. Similarly B is the part of J^T that does depend on L_4. This set of equations can be solved numerically using Newton-Euler iteration.

First, the differential relationship between F, L_4, and τ is derived giving:

$$\begin{pmatrix} A + BL_4 & BF \end{pmatrix} \begin{pmatrix} \Delta F \\ \Delta L_4 \end{pmatrix} = \Delta\tau. \tag{4.12}$$

This equation can then be applied directly to create an iterative solution given by:

$$\begin{pmatrix} A + BL_{4\,i} & BF_i \end{pmatrix} \left(\begin{pmatrix} F \\ L_4 \end{pmatrix}_{i+1} - \begin{pmatrix} F \\ L_4 \end{pmatrix}_i \right) = \Delta\tau - [A + BL_{4\,i}]F_i. \tag{4.13}$$

Because the iteration is a Newton-Euler scheme, if the system is going to converge, the convergence is quadratic [Strang 86]. The problem becomes one of choosing a good first guess for the solution. Experimentally, we have found that a good initial choice is $L_4 = 0$ for initial contact, or the value of the previously computed contact location if contact has been maintained. The initial estimate of F in either case is generated from the pseudoinverse of equation 4.11.

An examination of the error at each step shows that the iteration can stop under a number of conditions. The error is the difference between the actual torques and the torques that are calculated from the estimate of F and L_4. At step $i+1$ this is:

$$e_{i+1} = \tau - [A + BL_{4\,i+1}]F_{i+1}.$$

By using equation 4.11 to express τ as a function of F_{i+1} and $L_{4\,i+1}$ and the actual values of F and L_4, this can be expressed as:

$$e_{i+1} = [A + B(L_4 - L_{4\,i+1})](F - F_{i+1}).$$

This error can go to zero if 1) the solution converges to the correct solution, 2) if the projection of the error in the estimate of F into the space spanned by A is zero and L_4 converges to the correct result, or 3) if the projection of the force error into $A + B\Delta L$, or J^T, is zero. In order for condition three to hold, the rank of J^T must be equal to two. By performing Gaussian elimination, where each pivot is chosen such that it is always nonzero, we arrive at two relationships which must be zero simultaneously in order for the rank of J^T to be equal to two:

$$\frac{L_4(L_3S_2S_3S_4 + (A_3C_4 - A_4)S_2S_3)}{A_4} + A_3S_2S_3S_4 + (-L_4 - C_4L_3)S_2S_3 = 0 \quad (4.14)$$

$$\frac{L_4(C_3L_3S_4 + C_3(A_3C_4 - A_4))}{A_4} + A_3C_3S_4 - C_3L_4 - C_3C_4L_3 = 0. \quad (4.15)$$

The solutions for both equations can be intersected to find the conditions under which J^T will be rank two. Equation 4.14 will be zero if S_2 or S_3 is zero and equation 4.15 is zero if C_3 is zero. Additional possibilities are found by factoring these solutions out and solving for L_4 as a function of θ_4 for both equations 4.14 and 4.15. Both solutions result in the same equation:

$$L_4 = \frac{A_3A_4S_4 - A_4L_3C_4}{L_3S_4 + A_3C_4}. \quad (4.16)$$

The simultaneous satisfaction of these conditions results in two independent possibilities. If $C_3 = 0$ and $S_2 = 0$ the rank will be two and a force applied in the direction $[\,A_4 \quad 0 \quad -L_4\,]$ will result in no net torque on the manipulator. The corresponds to a force applied through joint 4 with the manipulator vertical and joint 3 turned 90°.

The second condition arises when equation 4.16 is satisfied. In this situation, forces applied in the direction $[\,L_3S_4 + A_3C_4 \quad 0 \quad C_4L_3 - A_3S_4\,]$ will result in no net torque on the manipulator. The location L_4 is the intersection of the axis of the second link and a line which passes through the origin's of frames 4 and 1. A force at this point along this line produces no torques on the manipulator. The problem will be ill conditioned and the iteration will not work when either of these possibilities occurs and makes J^T rank 2.

If only one of equations 4.14 or 4.15 is zero and ${}^5\hat{y}$ force is zero, J^T will still be rank 3, but two of the nonlinear constraint equations will become dependent. If $\theta_2 = 0$, $\theta_3 = 0$, or $\theta_3 = \pi/2$, the ${}^5\hat{x}$ and ${}^5\hat{z}$ forces affect only two torques. For the first two cases, torques are produced about joints 2 and 4, and in the second case torques are produced about joints 1 and 4. If the ${}^5\hat{y}$ force is zero two of the torque measurements will be zero. We can determine from these zero torques that this ${}^5\hat{y}$ force is zero but we cannot infer the contact location. Therefore, the two remaining torque measurements are insufficient to determine the ${}^5\hat{x}$ and ${}^5\hat{z}$ forces and the contact location. Additional information must be placed into the system and we have chosen the heuristic that the force in the ${}^5\hat{z}$ direction is zero.

With this procedure, we can use the manipulator as a sensor and can apply it to sensing collisions during a search. The algorithm can detect contact with an object and detect the motion of the contact along the link as the search proceeds. However, the procedure will not be able to resolve contacts at more then one location or contacts along the inner link. Both of these types of contacts will be incorrectly resolved by the algorithm as contacts along the last link. More sophisticated exploratory techniques and/or sensors would be required to resolve this problem.

4.2.4 Endpoint Inverse Kinematics

The MIT-WAM has a single redundancy in placing the endtip at a given location in cartesian space. This redundancy can be visualized by imagining that the endpoint of the manipulator is attached to a given point. It is then possible to swing the manipulator about the axis from the origin to this fixed point (see figure 4.3). Swinging the arm in

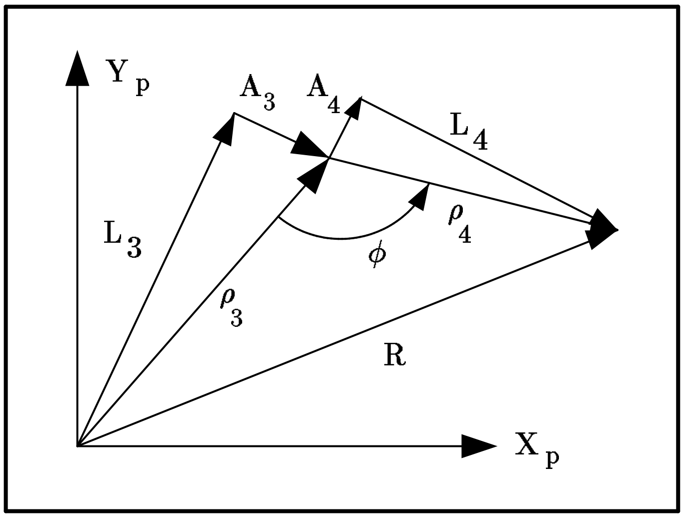

Figure 4.3: **Endtip Inverse Kinematics Elbow-up**. A pair of cones is formed by the links of the MIT-WAM if the endtip location is kept fixed and the first link is rotated about R. The distance vector from the base to joint 4 is ρ_3, and the distance vector from joint 4 to the end of the manipulator is ρ_4. This configuration is called the "elbow-up" configuration for the manipulator.

this way will cause the two arm links to sweep out cones.

This redundancy can be resolved by specifying an additional constraint. Possibilities include minimization of the effective endtip mass in the direction of motion, maximization of the workspace, minimization of the required joint torques, or fixing one joint angle. We choose to use the simplest constraint which is to solve for all of the other joint angles in terms of the desired position and θ_3.

Given a choice of θ_3, we can draw the manipulator in the plane formed by the manipulator's links. There are two possible configurations in this plane that will reach the desired location. These are shown in figures 4.3 and 4.4 and will be referred to as elbow-up and elbow-down, respectively.

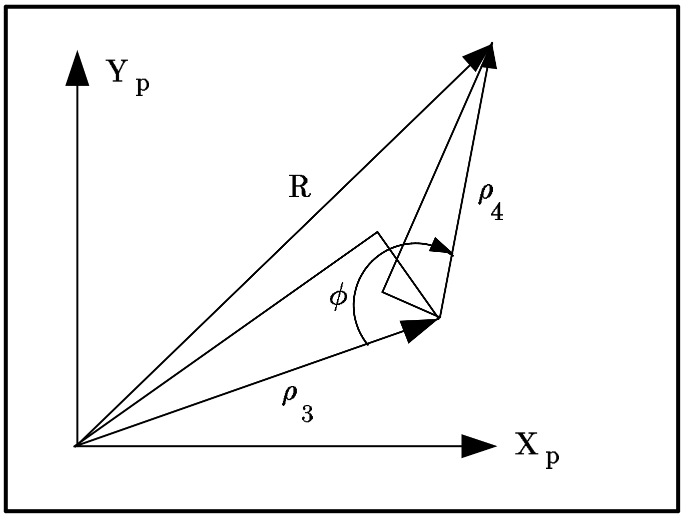

Figure 4.4: **Endtip Inverse Kinematics Elbow-down**. In this configuration ϕ is again an interior angle of the triangle formed by the two distance vectors ρ_3 and ρ_4.

From figures 4.3 and 4.4 it can be seen that vectors $\vec{\rho}_3$ and $\vec{\rho}_4$ sum to R, the vector from the origin of the base coordinate system to the desired position x. Because both of the links lie in a plane, θ_4 is determined by the magnitude of R to within two possible solutions. For both solutions ϕ, the interior angle to $\vec{\rho}_3$ and $\vec{\rho}_4$, can be determined by the law of cosines

$$\phi = cos^{-1}((\rho_3^2 + \rho_4^2 - r^2)/(2\rho_3\rho_3)),$$

where ϕ is chosen to be $0 \leq \phi \leq \pi/2$ since it is the interior angle of the triangle formed by ρ_3 and ρ_4,

where

$$|\rho_3| = \sqrt{{L_3}^2 + {A_3}^2} \tag{4.17}$$

$$|\rho_4| = \sqrt{{L_4}^2 + {A_4}^2}. \tag{4.18}$$

For the elbow-up configuration, θ_4 is given by:

$$\phi - \pi - \phi_4 - \phi_3 = \theta_4 \tag{4.19}$$

which will result in a negative number. For the elbow-down configuration, θ_4 is given by:

$$\phi - \pi + \phi_3 + \phi_3 = -\theta_4 \tag{4.20}$$

which will result in a positive number where:

$$\phi_3 = tan^{-1}(A_3/L_3), \quad \phi_4 = tan^{-1}(A_4/L_4). \tag{4.21}$$

In order to calculate the three base joint angles, the constraint that the origin of the last frame must be at $\rho_3 \hat{r}$ is used, but now with knowledge of θ_3 and θ_4. Using transform notation, we require that:

$$ {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{4.22}$$

Since θ_3 and θ_4 are known, this can be expressed as:

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ 1 \end{pmatrix} = {}^2_3T\,{}^3_4T\,{}^4_5T \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = {}^2_1T\,{}^1_0T \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{4.23}$$

where we have defined (α, β, γ) for convenience as the product of the transforms.[3] The constraints can then be reduced to three conditions on θ_1 and θ_2:

$$C_1C_2x + S_1C_2y + S_2Z = \alpha \tag{4.24}$$

$$-C_1S_2x - S_1S_2y + C_2z = \beta \tag{4.25}$$

$$S_1x - C_1y = \gamma. \tag{4.26}$$

[3]note: $({}^a_bT)^{-1} = {}^b_aT$. See [Craig 87].

By using the half-angle substitution,[4] equation 4.26 can be solved for θ_1

$$\begin{aligned} u_1 &= \frac{-x \pm \sqrt{x^2 + y^2 - \gamma^2}}{(y - \gamma)} \\ \theta_1 &= 2tan^{-1}(u_1). \end{aligned} \tag{4.27}$$

Equations 4.25 and 4.26 can be solved to determine θ_2 as a function of θ_1 as:

$$\theta_2 = \pm cos^{-1}(\frac{z\beta + (C_1 x + S_1 y)\alpha}{\alpha^2 + \beta^2}), \tag{4.28}$$

where the choice of sign for equation 4.28 is dictated by the choice of θ_1 in equation 4.28.

This procedure creates four solutions for each desired point. The solution closest to the current configuration is chosen as the desired solution.

4.3 Line Kinematics

As was shown earlier, line motion is a convenient parameterization that can be used to specify the motion of the manipulator performing the tasks described in chapter 2. A line is specified by four coordinates.[5] This can be seen by noting that for a general rigid body in three space, six coordinates are required to specify the location and orientation of the body, but for a line the translation along the line axis and rotations about the axis are unspecified. The line can be described by two points along the line, by a point on the line and a vector giving the direction of the line, or by the intersection of two planes. The first two provide a more convenient form for describing the position of manipulator links.

[4] $sin(\theta) = 2u/(1 + u^2)$, $cos(\theta) = (1 - u^2)/(1 + u^2)$, and $u = tan(\theta/2)$.
[5] This development follows [Shimano 78].

For the two point approach, the first point P_1 is described by its homogeneous coordinates $(\,x_1 \quad y_1 \quad z_1 \quad 1\,)$ and the second point P_2 is described by its coordinates $(\,x_2 \quad y_2 \quad z_2 \quad 1\,)$. The line can then be defined by the pair of points P_1 and P_2 as:

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \end{pmatrix}. \tag{4.29}$$

This set of terms can be changed to an alternative set by taking the second-order determinate. The new expressions are:

$$\begin{aligned} P_{01} &= x_2 - x_1 \\ P_{02} &= y_2 - y_1 \\ P_{03} &= z_2 - z_1 \\ P_{23} &= y_1 z_2 - y_2 z_1 \\ P_{31} &= z_1 x_2 - z_2 x_1 \\ P_{12} &= x_1 y_2 - x_2 y_1. \end{aligned} \tag{4.30}$$

These quantities are called the Plücker or line coordinates of a line. The first three components are sometimes grouped into a vector $\hat{l}$ which can be identified as a vector in the direction of the lines axis. The second group of three components can be grouped into a vector $\tilde{l}$ which can be identified as the cross product of the vectors from the origin of the coordinate system to each of the points.

As was mentioned earlier, a line is specified by four independent parameters. The two extra freedoms in the Plücker coordinates come from the fact that different pairs of points on the line could be chosen to describe the same line. Choosing two different points changes the scaling of the coordinates. The Plücker coordinates can be normalized by dividing $\hat{l}$ and $\tilde{l}$ through by $|\hat{l}|$. The new normalized coordinates will be designated by $\hat{L}$ and $\tilde{L}$. $\tilde{L}$ is often termed the moment of the line because it has the magnitude and direction of the moment produced about the origin by a unit force acting along the line. $|\tilde{L}|$ can also be seen to be the shortest distance from the line to the origin of the system. The second constraint relationship comes from the fact that $\hat{L}$ and $\tilde{L}$ are perpendicular. This can be proven by taking the dot product of the two vectors.

The next sections use a two point line specification and the associated Plücker coordinates to derive solutions for the forward and inverse kinematics of a line for the MIT-WAM. Lastly, the Jacobian of a line is introduced and derived based on transform techniques.

4.3.1 Forward Kinematics

The forward kinematics for lines is the problem of calculating the normalized line coordinates for the line through the center of the last link. After transforming, the line coordinates of the last link can be compared to the desired line specified by a the line trajectory generation scheme developed in section 4.4. Because the location of a line is easy to visualize by specifying two points, the trajectory generation schemes use a two point technique to specify the desired line motion.

The line forward kinematics is the problem of computing the Plücker coordinates of the line through the center of the last link given the joint angles. The direction of the line is computed by transforming the unit vector along the last link (z_5 See figure 4.2) into the base coordinate system by premultiplying with the transform matrices. The moment of the line is found by taking the cross product of a unit vector in the direction of the line, with a vector from the origin to a point on the line. For the manipulator it is convenient to use the cross product of z_5, transformed into the base system, and the vector from the origin to the origin of frame 5 expressed in the base coordinate system in order to compute $\tilde{L}$. These relationships determine the line through the center of the last link.

$$\hat{L} = {}^0_5T z_5 \quad \text{is the direction of the line,} \tag{4.31}$$

$$\tilde{L} = O_5 \times {}^0_5 T z_5 \quad \text{is the moment of the line, and} \tag{4.32}$$

$$\text{where } O_5 = {}^0_5 T \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T .$$

4.3.2 Inverse Kinematics

One of the concepts for whole-arm manipulation is to use all of the robot's surfaces to produce useful motions. As a first step towards this goal, we consider the task of placing the last link along a line in space. Given an analytic solution for the joint coordinates to place the link along a given line, we then translate the line in directions normal to the line to cause the manipulator to sweep out planes. These sweeps can be used to search for objects, to push objects along a trajectory normal to the line and to move objects into graspable configurations.

The inverse kinematics for the line geometry is solved by a series of geometric transformations. First the Plücker coordinates are formed from the two points, x_1 and x_2, which describe the line. Two cases can now be identified. If the magnitude of $\tilde{L}$ is non-zero, the desired line does not pass through the origin, and consequently the two points

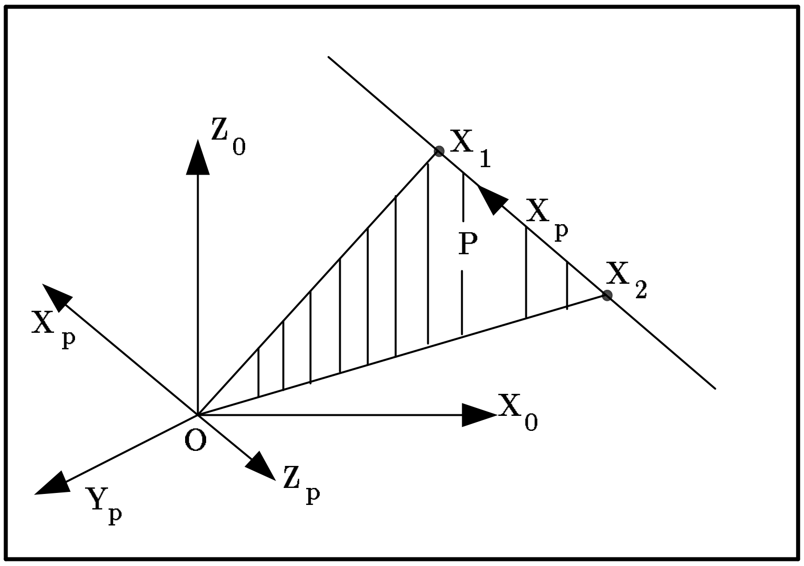

Figure 4.5: **Line Geometry and Definition of** P. Note that X_1 always lies in the positive X_p direction.

and the origin define a plane. In the second degenerative case, the line passes through the origin and one constraint on the solution is lost. We first examine the general case and then show how the procedure can be applied to the degenerative case.

Figure 4.5 shows the plane defined by x_1, x_2, and the origin O. Let a new right-hand plane coordinate system be defined by $\hat{L}$, $\tilde{L}$, and the cross product of these terms. This produces three new unit vectors which we will label $\hat{x}_p$, $\hat{y}_p$, and $\hat{z}_p$. These unit vectors are computed by:

$$\hat{x}_p = \quad \hat{L} \quad = \frac{x_1 - x_2}{|x_1 - x_2|} \tag{4.33}$$

$$\hat{z}_p = \quad \tilde{L}/|\tilde{L}| \quad = \frac{x_1 \times x_2}{|x_1 \times x_2|} \tag{4.34}$$

$$\hat{y}_p = \quad \hat{z}_p \times \hat{x}_p. \tag{4.35}$$

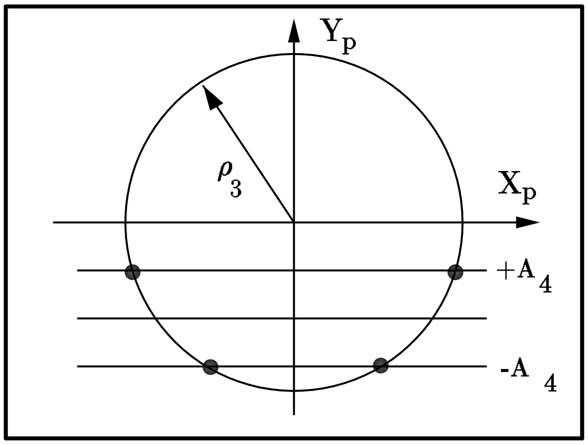

Figure 4.6: **Locus of Possible Joint 4 Locations**. For a given desired line two translated lines a distance A_4 away are formed. Joint 4 must lie on the intersections of these translated lines with a circle of radius ρ_3.

Equations 4.33 through 4.35 define a rotation matrix ${}^p_0R = [\hat{x}_p \quad \hat{y}_p \quad \hat{z}_p]$ which takes x_1 and x_2 into a coordinate system such that x_1 and x_2 are determined by $\hat{x}_p$ and $\hat{y}_p$ only. Let the coordinates of x_1 and x_2 be redefined in the new system by $({}^px_1, {}^py_1)$ and $({}^px_2, {}^py_2)$. Since the $\hat{x}_p$ direction is the direction along the line and both points lie on the line at a distance $|\tilde{L}|$ from the origin, ${}^py_1$ and ${}^py_2$ must be equal to $-|\tilde{L}|$.

In order for the last link to lie on the desired line, joint 4 must lie along a line offset from the desired line by A_4. Two such offset lines are possible. Since joint 4 must also lie along a circle of radius ρ_3, the possible locations of joint 4 are generated by intersecting the circle and the two offset lines. Figure 4.6 shows the geometry of the possible solutions in the plane P. Depending on the distance from the line to the origin, there can be zero, one, two, three, or four intersections. For each intersection, the last link can be placed on the line in two directions. By requiring the last link to lie in the direction of $\hat{x}_p$, or

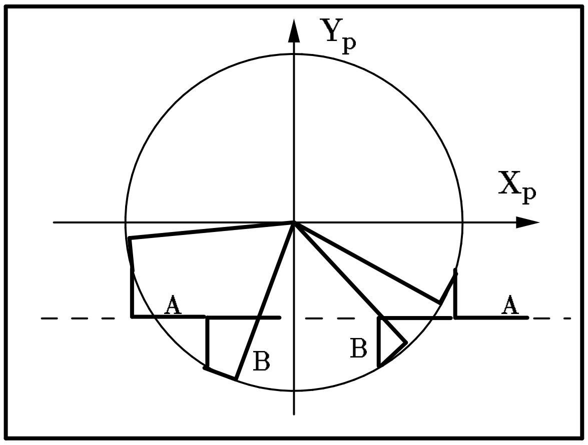

Figure 4.7: **Configuration of Solutions** There are four possible configurations of the robot that will place the link on the desired line. Two solutions, labeled A, lie at at the intersection of the circle with the line translated by A_4. The other two, labeled B, lie in at the intersections with the line translated by minus A_4.

in the direction from x_2 to x_1, one of these solutions is eliminated. Figure 4.7 shows the geometry of the four possible solutions.

The four θ_4 solutions can be grouped into two different configurations which are labeled A and B in figure 4.7. The A solutions correspond to the crossings from the addition of A_4 to the desired line and the B solutions correspond to the subtraction of A_4 from the desired line. A detail of the A configuration is shown in figure 4.8. From the figure, the value θ_4 can be found directly to be:

$$\phi - \phi_3 = \theta_4 \tag{4.36}$$

$$\phi = sin^{-1}(y_p/\rho_3). \tag{4.37}$$

The interior angle of link3 is defined as $\phi_3 = tan^{-1}(A_3/L_3)$. The origin of frame 3 lies at the end of link 3 and is at position $[\,L_3C_4 \quad L_3S_4 \quad 0 \quad 1\,]$ in the plane P. We require that

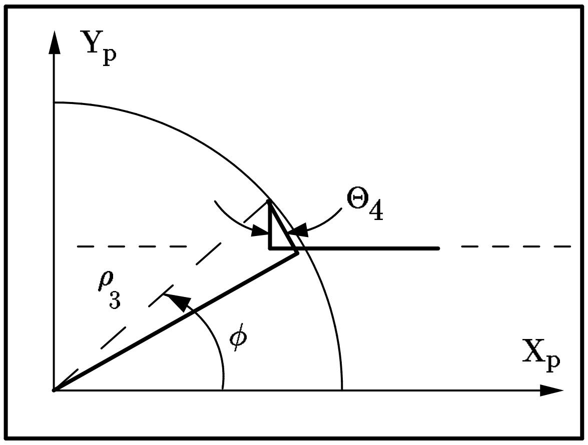

Figure 4.8: **Configuration A** The figure shows the geometry of the solutions for the line translated by $+A_4$.

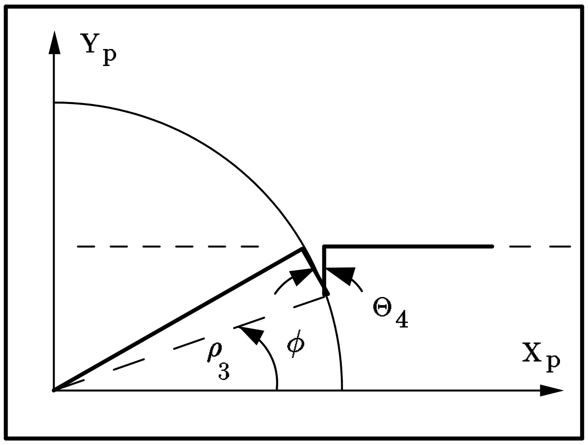

Figure 4.9: **Configuration B** The figure shows the geometry of the solutions for the line translated by $-A_4$.

θ_1 and θ_2 be chosen to locate the origin at this desired location

$$\begin{bmatrix} {}^0_pR \begin{bmatrix} L_3C_4 \\ L_3S_4 \\ 0 \\ 1 \end{bmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} l \\ m \\ n \\ 1 \end{bmatrix} = {}^0_3T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.38}$$

We have defined $[\,l \quad m \quad n\,]$ for convenience. This constraint becomes:

$$\begin{bmatrix} -C_1S_2L_3 \\ -S_1S_2L_3 \\ C_2L_3 \end{bmatrix} = \begin{bmatrix} l \\ m \\ n \end{bmatrix}. \tag{4.39}$$

This is solved for θ_1 and θ_2 as:

$$\theta_2 = \pm cos^{-1}(n/L_3) \quad \theta_1 = tan^{-1}(m/l), \tag{4.40}$$

producing two solutions for each θ_4. Lastly to solve for θ_3 in configuration A, we require that z_4 be aligned with $-z_p$

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = {}^0_p R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = {}^0_4 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.41}$$

where $[\alpha\beta\gamma]$ are defined for convenience.

This constraint specifies the tangent of θ_3 by:

$$C_3 = C_1\beta - S_1\alpha \tag{4.42}$$

$$S_3 = -C_1\alpha - S_1\beta \tag{4.43}$$

$$\text{or, } S_3 = \gamma/S_2 \tag{4.44}$$

$$\theta_3 = tan^{-1}(S_3/C_3). \tag{4.45}$$

The resulting procedure produces two unique solutions for configuration A. For configuration B, two requirements change. The last joint angle is found from figure 4.9 as:

$$-\phi - \phi_3 = \theta_4. \tag{4.46}$$

This is used in equation 4.38 and the results applied to equations 4.40 and 4.40 to solve for joint angles θ_1 and θ_2. Lastly to solve for θ_3 in configuration B, we require that z_4 be aligned with z_p. This results in equations 4.45 with the signs reversed. With this procedure we can produce all of the solutions for the planar case. There can be at most four solutions and the closest solution to the current configuration is the chosen solution.

For the degenerative case, the plane P is not defined and we must introduce an additional constraint. It becomes possible to place the last link along the line with any value of θ_3. For convenience the current implementation is to use the current value of θ_3 and thus define z_p by z_4. Once the plane is defined the non-degenerative procedure can be applied to solve for the joint angles.

4.3.3 Line Jacobian

Although the absolute position of the line is best discussed with Plüker coordinates, it is not clear how that relationship should be applied to differential motion of the line. For control, we need to be able to apply a corrective torque to the joints which opposes forces defecting the line from the desired location. The line can deflect along and rotate about two axes normal to the line. To determine the rotations and deflections, we measure the motions relative to some frame of reference. It is for this reason that we introduce the line

Jacobian relative to a frame fixed to the line at a location specified by the programmer.

This frame would, in general, define six relationships (three translations and three rotations for each axis), however with a line, translation along the line axis and rotation about the line axis is undefined. The Jacobian then relates the joint angular velocities to the generalized velocities of the four remaining parameters. This relationship is used in the stiffness control algorithm described in chapter 3. The line Jacobian consists of the first two rows of the Jacobian of section 4.2.2, which relate the joint velocities to the x_5 and y_5 linear velocities, and rows four and five which relate the joint velocities to the x_5 and y_5 rotations. The relationship is restated here:

$$\begin{aligned}
J_{1,1} &= A_3S_2S_3S_4 + (-L_4 - C_4L_3)S_2S_3 \\
J_{1,2} &= A_3C_3S_4 - C_3L_4 - C_3C_4L_3 \\
J_{1,3} &= 0 \\
J_{1,4} &= -L_4 \\
J_{2,1} &= (A_4C_3S_2 - C_2L_4)S_4 + (-C_3C_4L_4 - C_3L_3)S_2 + C_2(A_3 - A_4C_4) \\
J_{2,2} &= (C_4L_4 + L_3)S_3 - A_4S_3S_4 \\
J_{2,3} &= -L_4S_4 - A_4C_4 + A_3 \\
J_{2,4} &= 0 \\
J_{3,1} &= C_2S_4 + C_3C_4S_2 \\
J_{3,2} &= -C_4S_3 \\
J_{3,3} &= S_4 \\
J_{3,4} &= 0 \\
J_{4,1} &= -S_2S_3 \\
J_{4,2} &= -C_3 \\
J_{4,3} &= 0 \\
J_{4,4} &= -1.
\end{aligned} \tag{4.47}$$

4.4 Trajectory Generation

The inverse kinematics are a component of a trajectory generation procedure used to move the manipulator in cartesian or line trajectories defined in the base coordinate system. We have implemented a robust joint trajectory executor which allow the user to command joint motions with prescribed stiffnesses. The high level software and the data structures used in our implementation are based on [Taylor 79]. The user is insulated from the implementation and is given a high level set of functions to insert, append, and copy trajectories into the system.

The implementation resolves line and cartesian motions into joint motions. A motion is created by planning a sequence of joint via points that satisfy a “closeness” constraint by using a recursive bounded deviation algorithm to refine the required path. The trajectory generator then interpolates between the via points using either a cubic or quintic spline interpolation scheme.

4.4.1 Joint Trajectories

Joint trajectories are specified by a series of points and delta times in which the manipulator is required to move from one point to the next. The list of points consists of start and end points, which have a desired velocity and acceleration associated with them, and a series of via points. The points can be generated by teaching joint motions, by entering a list of points in a file, or by passing on-line an internal buffer of points generated by the line or cartesian path planner.

Given a list of points, the generator calculates a set of coefficients for either a cubic or a quintic spline which will interpolate between points. The cubic spline scheme requires that the manipulator have a given position and velocity at the start and end point of the trajectory, that it pass though each via point, and that at each via point the velocities be continuous. A quintic spline has the additional requirements that the start and end accelerations are specified, and that at each via point the acceleration, jerk, and snap (the derivative of the jerk) must be continuous.

For example, between two neighboring points, a cubic spline is given by:

$$a_1 \Delta t^3 + a_2 \Delta t^2 + a_3 \Delta t + a_4 = \theta(t), \tag{4.48}$$

where Δt is the time to move from the first point to the next. To solve the constraint problem for n points, a banded linear equation is created for the $4(n-1)$ coefficients involved in the $(n-1)$ interpolation splines. The solution of this equation generates all of the coefficient values for the given trajectory. This set of values is then stored in the trajectory data structure and executed by the trajectory processor.

Software Structure

The trajectory program constitutes an extensive section of the controller software. The trajectory generator executes on a separate processor and is responsible for executing trajectories and generating joint trajectories from files and internal buffers. The software is written to make the execution of trajectories transparent to the user and to allow the user to interrupt and append trajectories at anytime during execution.

The trajectory system is based on a pair of linked lists. One list contains all of the trajectories that the robot currently knows how to execute. This list can be edited, appended, and queried. The second list is the executing list.

A trajectory is represented as a series of executable joint motion descriptor blocks. Each joint block has the structure:

```
struct _traject_entry{
        float position
        float velocity
        int type
        array poly
        }.
```

The position and velocity of each joint block specifies the state the joint is to reach at the end of an executing block. Type is the category of trajectory which can be a cubic spline, quintic spline, linear spline, or any other category for which an interpolation function has been provided. Currently, cubic and quintic splines are implemented. Poly is an array containing the coefficients of the interpolation scheme that take the joint from the previous block to the current block.

A manipulator trajectory is then created by building a linked list of trajectory blocks. Each trajectory block contains pointers to four joint blocks and the times associated with the manipulator block. The structure is:

```
struct _trajectory{
        struct _trajectory *next
        float end_time
        float delta_t
        array kp
        traject_entry *value[JOINTS]
        }.
```

Next is a pointer to the next block and implements the linked list, end_time is the absolute time at which this block is no longer valid, delta_t is the duration for which this block is valid and is used in the interpolating scheme, kp is the stiffness matrix (4x4) which is to be used during the current block, value is a list of pointers to all of the joint blocks for the current manipulator block. Each linked list of this type is called a trajectory. In the system, a list of executable trajectories is maintained. This list can be queried, edited, or copied into the executing list.

The executing list can only be changed by copying in new trajectories or by appending or inserting new trajectory blocks. Execution of a trajectory is accomplished by first initializing the trajectory which builds a coefficient array to take the manipulator from its current location to the start of the trajectory. Execution of the trajectory proceeds by running a servo loop and calling the function:

push_traj_to_servo(time, des_pos, des_vel, des_acc).

The user of the trajectory system provides the system with pointers to the desired position, velocity, and accelerations of the joint and the current time. The system searches the time ordered trajectory list for the first block with an end_time greater that the current time, starting from the last block which was executed, and then uses this block to calculate and update the desired position, velocities, and accelerations for each joint.

The ability to interrupt executing trajectories is used to implement guarded moves. To perform a guarded move, the manipulator moves in a specified direction until contact occurs. Upon contact, it interrupts the running trajectory and inserts a trajectory that has a desired position equal to the current position, thus stopping the manipulator. To smooth the insertion, the trajectory system inserts a transition block from the current location to the beginning of the new block. Thus, the high level code for a guarded move is conceptually clean:

```
move_until_contact(time,delta_joint1,delta_joint2,delta_joint3,delta_joint4)
{
        int ret
        set_contact_flag()
        move_by_internal(time,delta_joint1,delta_joint2,delta_joint3,delta_joint4)
        if((ret = return_on_contact()) == CONTACT)
                stop_motion()
        return ret
}.
```

4.4.2 Line and Cartesian Trajectories

In addition to joint trajectories, the system generates cartesian and line trajectories. Both of these trajectories are created by performing the inverse kinematics on the desired points and writing the resulting joint angles to either a file or an internal buffer. The trajectory generation scheme then takes the file and builds a joint trajectory to follow the desired path. In order to track the desired motion the method of bounded deviation paths [Taylor 79] has been implemented for both cartesian motions and line motions

The user provides the algorithm with a maximum deviation bound, which is used to recursively refine the generated joint motions to keep the joint trajectory within the desired bound. For a cartesian trajectory the bound is a the maximum distance from the path that the manipulator may deviate. For a line trajectory, two numbers are provided to calculate three bounds. The magnitude of $\tilde{L}$ must be within a distance bound. The angle between the desired and actual $\hat{L}$ must be within and angles bound, and the angle between the actual and desired $\tilde{L}$ must be within the same angle bound.

The algorithm first calculates an inverse kinematic solution to the desired point. Next, the algorithm calls a function which checks to see if the joint motion from the previous joint location to the new joint location will stay within the desired bounds.

To do this, the joint angle half-way between the previous and new location is calculated. This angle is then transformed into a cartesian location and compared to the desired cartesian location half-way between the previous point and the current point. If the half-way point calculated resulting from the joint motion is within the tolerance band of the desired half-way point, the function returns true, otherwise it returns false.

The recursion then, branches on the returning value. A true return causes the recursion to write out the calculated joint angles and to return. A false return causes the recursion to generate a new desired point half-way between the previous point and the current point. The recursion then, calls itself with this new point.

After this new recursive call returns, the recursion will have calculated all of the joint angles up to the half-way point. The recursion then, goes back to the tolerance branch and tries to move from the half-way point to the current point. This is the complete recursion. The algorithm is sketched in code below:

```
recursive_inverse_kinematics( time, current_point,
previous_point, previous_angles, tolerance)
{
        find_inverse_kinematic_solution(current_point, new_angles)
compute:
        if(closer_than_tolerance(current_point, previous_point, new_angles,
        previous_angles, tolerance)) {
                write_out_joint_angles()
                return
        }
        else {
                find_mid_point(previous_point, current_point, mid_point)
```

```
                    time = time/2.0
                    recursive_inverse_kinematics( time, mid_point, previous_point,
                    previous_angles, tolerance)
                    }
            goto compute
    }.
```

There are two refinements to this recursion. First, near singularities the algorithm may not converge to a series of points which will satisfy the tolerance requirement. The function which checks the tolerance detects this condition and returns true when it occurs. Second, the velocity between joint via points must be bounded. This is accomplished by computing the maximum velocity that would occur for a straight-line trajectory from the previous joint angles to the current joint angles. The time, which is written to the output file, is adjusted to make the maximum velocity less than a predetermined maximum.

4.5 Conclusion

The motion of the MIT-WAM manipulator can be described by the joint coordinates, the cartesian location of the endpoint, or by the motion of the line through the last link. All of the relationships are important. Since the stiffness control algorithm, requires that all of the kinematic motions of the arm be specified by joint motions, we have developed a program for computing the inverse kinematics for a line or cartesian motion that will be close to the desired motion once translated into joint coordinates.

The differential, or Jacobian, relationships in this chapter can be applied to the generation of compliant motions in different spaces. For an end-effector, a compliance in the cartesian space of the end-effector might be used. For a WAM task, a line stiffness may be more convenient; consequently we have derived the line Jacobian.

A whole-arm task requires a technique for sensing the location of contact on the links. One possible technique is to use the joint torques, due to contact, to compute the contact location and forces. By making a few simple assumptions, we have shown that the contact forces and location can be computed with an iterative scheme.

These functions are applied to WAM motions and trajectory following in Chapter 6. Chapter 5, first describes the implementation of the kinematics and control strategies discussed in this and earlier chapters.

It should be noted that our treatment of line motion implicitly assumes frictionless line contacts (see [Mason and Salisbury 85]). This type of contact can transmit no force along the line's axis. In fact for our manipulator, friction does exist along the line contact. It remains to be investigated what limitations this will impose on our approach.

Chapter 5

Manipulator Control Implementation

5.1 Computational Hardware and Development Software

The computational hardware and architecture for the MIT-WAM arm is an extensible and powerful computational engine. An implementation of a robot controller for our application requires sufficient computation for control, including the possibility of complex adaptive controllers, on-line kinematics and trajectory planning, sensor fusion, and eventually high level planning. In order to accomodate the demands of the computational task, we have based our system on five 68020 single board computers acting in parallel and on the Condor software development environment developed by the UTAH-MIT hand research group [Narasimhan 88].

5.1.1 Hardware

The real-time control hardware for the MIT-WAM arm consists of five Ironics 3201 single-board computers, an Ironics 3273 Bus Controller, a Data Translation 1401 analog-to-digital converter, a Data Translation 1406 digital-to-analog converter, two Motorola MVME-340 parallel ports, and a HVE-2000 VME-VME bus-to-bus adapter, all housed in a VME-Bus card cage. In addition, four optical isolation boards and four Moog motor controllers are used to operate the robot. A SUN-3-280 workstation is connected to the real-time system through the bus-to-bus adapter. The Sun is used for program development and for plotting data stored on the Ironics processors.

The Ironics 3201 is based on a Motorola 68020 micro-processor with a 68881 floating point accelerator. The board contains one megabyte of RAM. All of the RAM is dual-ported; i.e. it can be accessed locally or across the VME bus.

The Ironics 3273 is used to control access to the bus. The Hal Versa Engineering (HVE) 2000 board connects the real-time control bus to the Sun-3. It provides a completely transparent connection between the Sun and the real-time bus allowing the Sun to access the memory of the micro-processor boards. The Data Translation 1406 D/A provides eight, 12 digitial-to-analog converters which are used to apply commands to the motor controllers. The Data Translation 1402 A/D has 32 single-ended analog-to-digital converters with 12-bits of resolution and programmable gains of 1, 2, 4, or 8.

The Motorola parallel-port provides direct digital I/O for the system. The board is used to read the digital position signals from the motor controllers without handshaking. While the board can be configured to provide I/O via interrupts, we are currently use

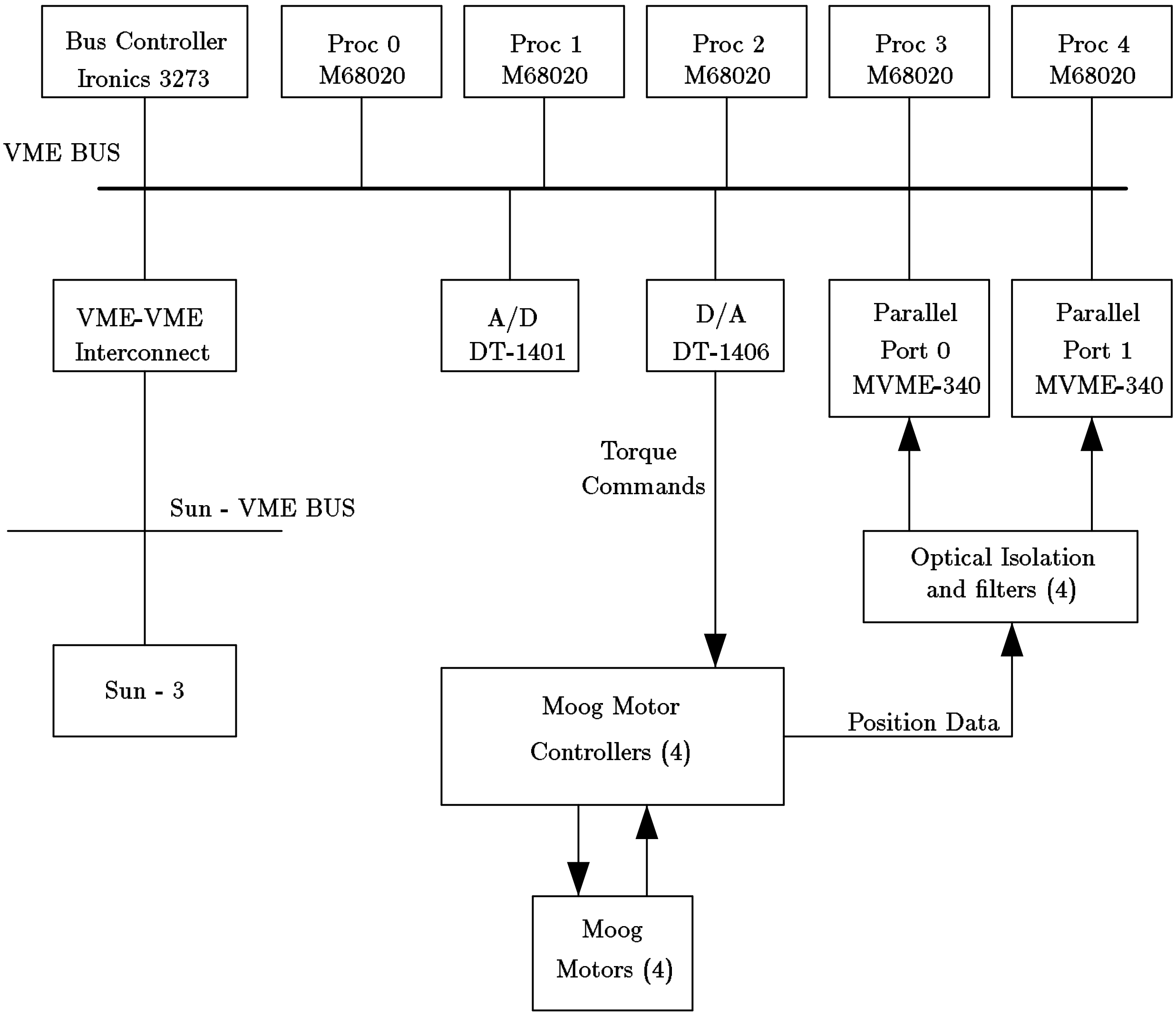

Figure 5.1: **MIT-WAM Control Hardware Block Diagram**

polling.

The optical isolation boards were built to interface the position data from the motor controllers to the parallel ports. The isolation boards are necessary because of the high levels of noise associated with the pulse width modulated (PWM) amplifiers used in the Moog motor controllers. The motor controllers and the isolation boards are described further in section 5.2.

5.1.2 Condor Software Support

The Condor system is a real-time software development environment tailored to the computational hardware used in the real-time controller. The Condor provides device drivers for the hardware components, a mailbox system for message passing, basic UNIX functions, debugging support, a virtual terminal for each processor, and a large number of user utilities.

The robot control program is split into separate programs, one for each processor in the system, and compiled on the Sun workstation. Since both the Sun and the Ironics processors are based on the 68020, a cross-compiler is not necessary. The compiled program is then linked to a real-time library containing UNIX functions and software utilities. Each of the resulting control programs is then downloaded to the micro-processors over the HVE-2000 to run. A virtual terminal connects each of the processors to the Sun so that output from the micros is printed directly on the workstation display.

The processors and the Sun can communicate either through the dual-ported memory on each processor, or through a software mailbox system. The mailbox system allows each processor to send a mail message, one piece of integer data, to any other processor. By using the mailbox system, the programs on each of the processors can be developed using the object oriented programming style. This keeps the level of complexity of interaction for the multi-processor system at a tractable level. However, it should be noted that for high-speed and large quantities of data, the mail-box system is not efficient and the dual-ported memory must be used directly. The control software for the MIT-WAM system uses a mixture of both to interface the processors.

The Condor system also provides a large number of user utilities. Functions to set timing interrupts, run servo loops, manipulate arrays, plot data, and to perform file serving are provided.

5.2 Motors and Controllers

The manipulator is actuated by four Moog 300-003 DC brushless motors controlled by four Moog 152P227 brushless torque controllers. The Moog motors have eight Samarium Cobalt motor poles (4 pole pairs) on the motor rotor. The stator is a 3 phase wye connection. A brushless resolver at the end of the rotor shaft provides position feedback for servo control and commutation. The motor is capable of applying 15 in-lb of continuous torque at speeds up to 7500 rpm. In addition, the motor can be pulsed to provide a peak torque of 60 in-lb.

The motor controller has the general form of figure 3.6. Each motor phases is driven by a PWM voltage amplifier at 320 volts switching at 5kHz. The three amplifiers can apply continuous power of 5 kW to the motor. A resolver to digital converter (R/D) provides motor shaft position with 12 bits of resolution. The motor controller determines the

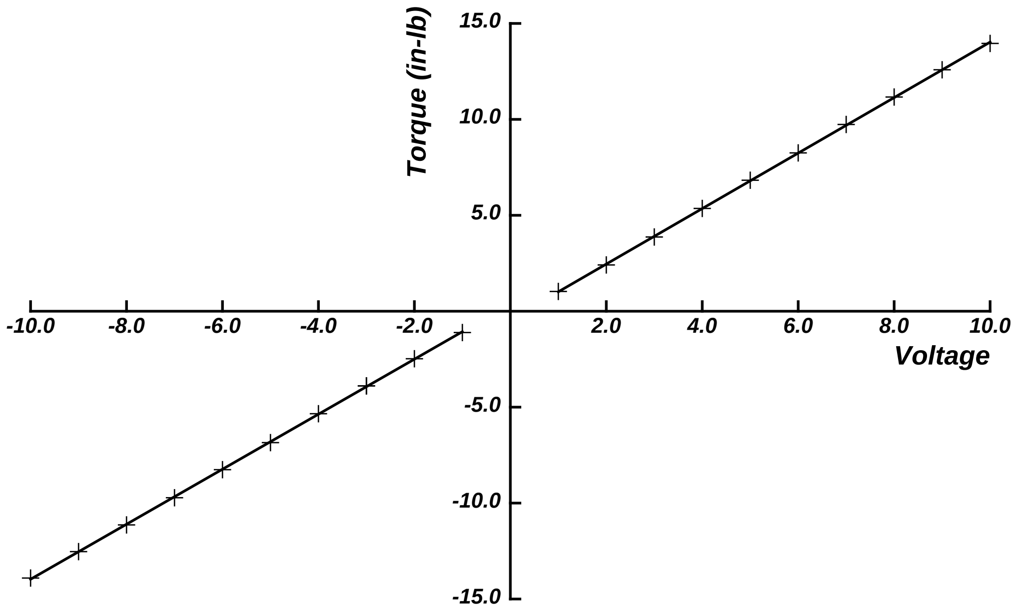

Figure 5.2: **Applied Motor Torque versus Voltage Command**. Note the deadzone in the output torque for small commanded voltages.

motor torque by closing a current feedback loop on each of the motor phase. Torque commands to the controller are scaled such that a 10 volt command produces 15 in-lb of torque output.

5.2.1 Deadband, Friction, and Ripple Measurements

The motor torque was measured using a commercial force sensor. The force sensor was mounted on a rotary stage concentric with the motor shaft. An aluminum beam mounted on the motor shaft, transmits the motor torque to the sensor. By rotating the force sensor slowly on the stage, the motor torque was measured as a function of motor position.

The torque to voltage characteristics, motor deadzone, and motor friction were obtained by averaging all the measured torques in the clockwise direction and in the counter-

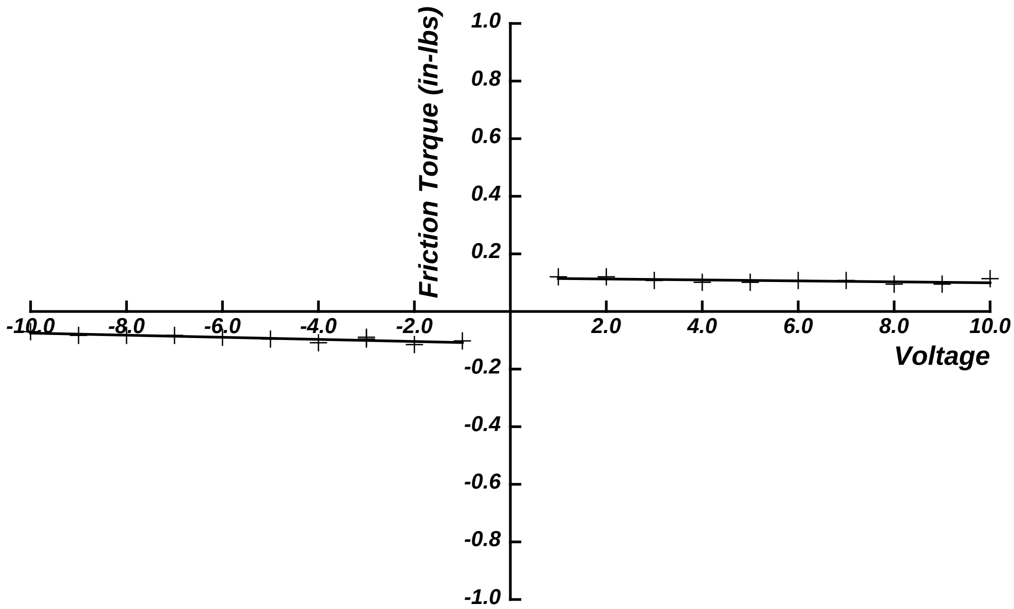

Figure 5.3: **Motor Friction versus Voltage Command**. The motor friction is 0.1 in-lb in either direction of rotation.

clockwise direction. The motor torque was calculated by slowly rotating the sensor about the motor, so that inertia effects could be neglected, and sampling the measured force for each direction of rotation. The clockwise and counter-clockwise torques were averaged to compute the applied torque. The difference between the commanded torque and the actual torque is the effective motor friction. A plot of the applied torque versus the command voltage is given in figure 5.2. Figure 5.3 shows the motor friction as a function of command voltage.

The total friction for joint four of 0.45 in-lb, and similarly for the other joints, can be broken into three components. Figure 5.2 shows that the motor controller has a command deadzone of approximately 0.20 volts. This translates to an effective friction of 0.3 in-lb.

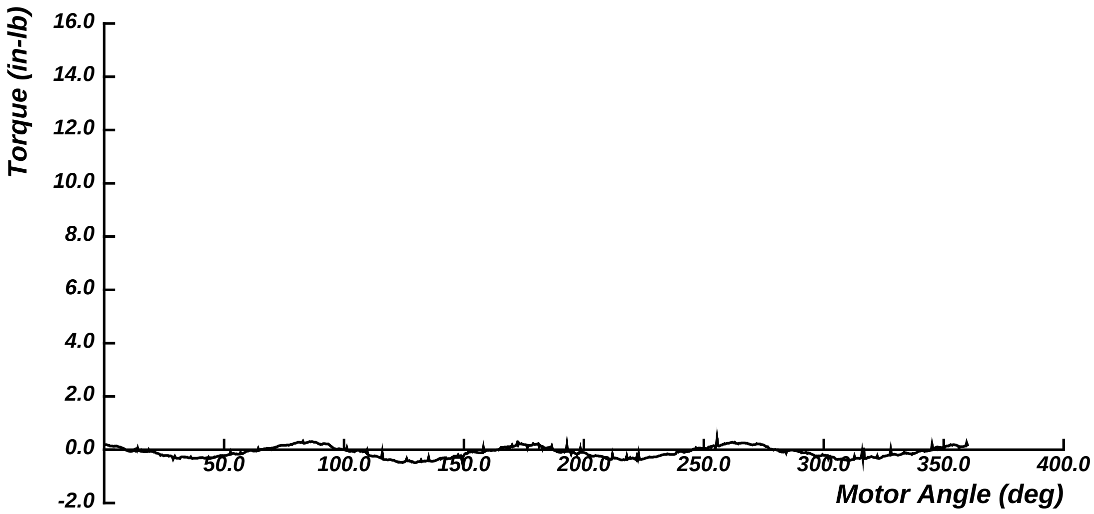

Figure 5.4: **Motor Torque versus Motor Angle for a command of 0 volts.**

The second friction term, motor friction, is 0.1 in-lb in either direction of rotation and is fairly independent of the command voltage. The remaining friction of approximately 0.05 in-lb is contributed to friction in the joints bearings and transmission losses.

A torque versus motor position plot for a command of 0 and 9 volts is shown in figures 5.4 and 5.5. The ripple torque is predominately at a frequency of four cycles per motor revolution in both figures. The magnitude of the ripple varies with the voltage command. The ripple that exists for a zero voltage command is termed the motor cogging, which has a constant magnitude and phase for all values of command voltage. The additional ripple scales linearly with the command voltage with a phase independant of voltage for low motor speeds. A plot of the fourth harmonic ripple magnitude as a function of command voltage is shown in figure 5.6.

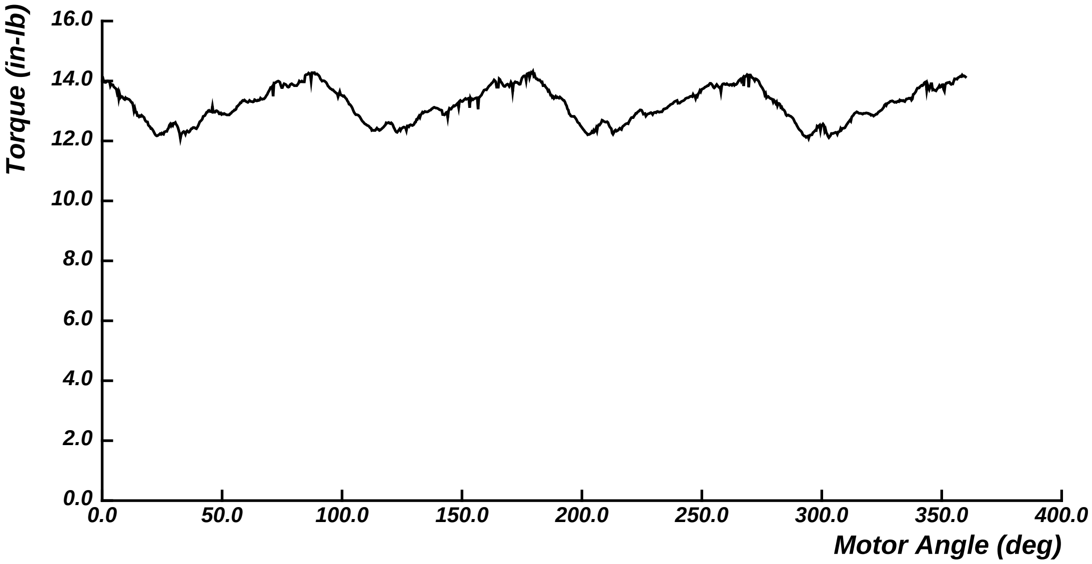

Figure 5.5: **Motor Torque versus Motor Angle for a command of 9 volts.**

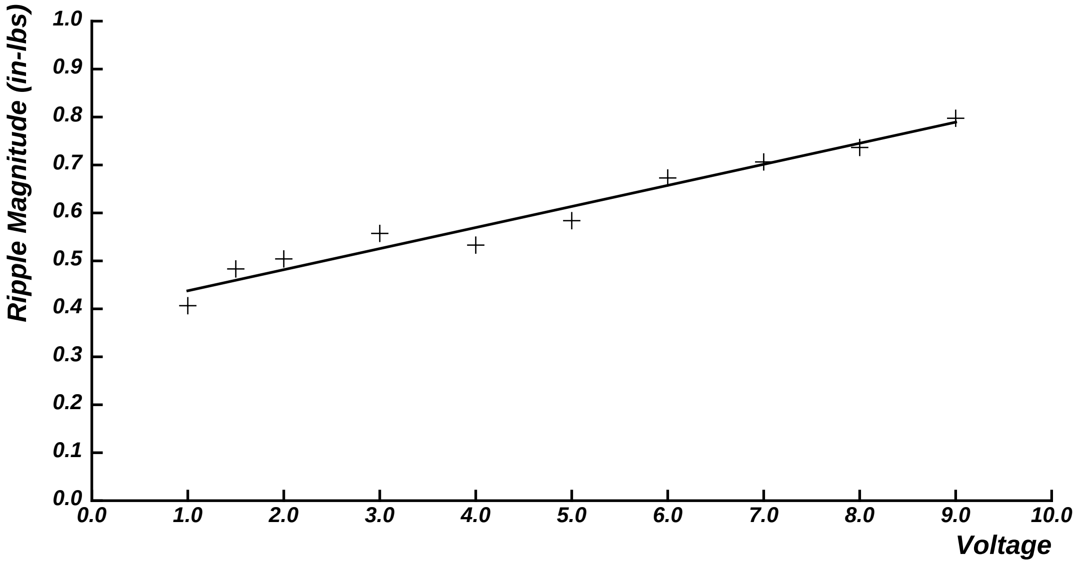

Figure 5.6: **Fourth Harmonic Ripple Magnitude versus Voltage Command.**

5.2.2 PWM Motor Noise

Electrical noise is frequently a limiting problem in any real-time control implementation. Considerable effort in this project was spent trying to control the noise induced by the PWM motor controllers. The PWM amplifiers apply square waves to the motors of 320 volts at 5 kHz. The rise and fall of each square wave produces noise spikes with a frequency of approximately 2 MHz. These spikes appear on any electrical equipment near the controllers or the robot. In fact, measurements of the noise is complicated by the introduction of the noise into the oscilloscope through the power plug. The noise is on all of the Moog controller components including the position outputs.

To reduce the effect of the noise, the grounding paths of the controllers and robot were carefully constructed. Grounding planes were provided by 1000 volt welding cable and brass bars so that the ground resistance within the controller cage would be very low. Each of the components was isolated from the cage with nylon washers so that the only ground path was through the cable and bars which helped to reduce ground loops.

Shielded cables were used in the power and signal lines; all of the shields are grounded at the controller cage. Where it was possible, twisted-shielded-pair cable was used on the signal lines. The Sun cage was physically separated from the controller cage by 8 ft, which was about the maximum given the size of the room. This was necessary to reduce the noise induced into the Sun cage.

The noise on the controller position outputs was the most problematic. A direct connection of the position outputs to the parallel ports resulted in unusable position data. This problem was solved by building a circuit board that uses optical isolation and filters to separate the ground planes of the controllers from the real-time system. The output of the optical isolators is an internal light signal generated by the common-mode voltage difference at the input to the isolator. The light signal modulates an internal phototransistor which is powered by separate voltages. The circuit uses HP 2631 high speed, high common-mode rejection, optocouplers.

In addition to the optical isolation, an RC filter with a cut-off at 1 MHz was necessary after the output of the optical isolators. We found that when the output of the optical isolator was high, high frequency transitions to ground occurred at a rate of 5 kHz. At the optical isolation boards, the noise on the ground and signals is no longer in perfect phase. Because of the difference in phase, the common-mode rejection is not complete; hence spikes occur. The addition of an analog RC filter on the optical isolator output removed this problem. At the maximum motor velocity of 7500 rpm, the lowest bit of the resolver is a square wave with frequency 256 kHz. Even at high motor velocities, the RC filter does not significantly delay this bit.

5.3 Software and Control Algorithms

The robot controller is based on five micro-processors. Each processor has a separate control function and program. The programs communicate through message passing for low bandwidth sporadic messages and through direct memory access for high bandwidth servo tasks. The control program is broken into an input program, an output program, a servo controller, a trajectory generator, and a task program and user interface. The following sections sketch the programs that run on each of the processors and the memory relationships that exist between them.

5.3.1 Input Processor

Processor three performs the input function for the control system. This processor polls the parallel ports for all the current motor positions. It converts the motor positions to an absolute position by tracking motor revolutions. The absolute position is filtered as an integer to smooth sudden transitions that can occur with the resolver. The filtered positions are then used to calculate the velocity of the motor by taking a first order difference. The resulting velocity is also filtered.

The R/D converter in the Moog controller performs a succession of internal A/D conversions to convert the analog resolver signals to a digital signal. While this is occurring, the resolver signal is not valid and a converter busy signal is asserted to indicate this condition. Because of the noise in the system, we were unable to incorporate the converter busy signal into the design of the resolver interface cards. Therefore, the parallel ports occasionally pass invalid data to the input processor. This problem is corrected by maintaining a circular buffer containing the last two position readings and the new reading. The median value of the three readings is accepted as the correct position value. The median value is then filtered by a single pole at 700 Hz. The filtered value is used to calculate the velocity with a first order difference which is then filtered by a single pole at 210 Hz. The input processor runs at 2000 Hz to minimize time delay effects in the filters. The resulting values of motor position and velocity are stored in local memory.

5.3.2 Output Processor

Processor four performs the output function for the control system. The processor receives a desired motor torque expressed as an output voltage from the servo controller. It then adds on the ripple compensation torque and the friction compensation torque. The total resulting torque is written to the D/A.

To compute the ripple compensation, the current motor position is required. The processor obtains this value by dereferencing a pointer to processor three's memory. The ripple compensation is then computed by looking up a feedforward term in a table. In order to compute the friction compensation, as described in section 3.3.1, the processor requires the desired position in motor coordinates in addition to the actual motor position. The desired position in joint coordinates is computed by the trajectory processor and the result is placed in the memory of the servo processor. The servo processor converts the joint desired position to motor coordinates and places this result in the the output processor's memory. The output servo runs at 1.5 kHz. This rate comes from the bandwidth requirements for ripple compensation (see section 5.3.2) and the requirements on switching frequency (see section 5.3.2).

Ripple Compensation

To compensate for the torque ripple, two tables are created from the values of the magnitude and phase of the cogging and torque ripple. The tables contain a value for every possible resolver angle (4096 points). To offset cogging, the servo controller adds the value of the cogging for the current motor angle to the commanded torque. To com-

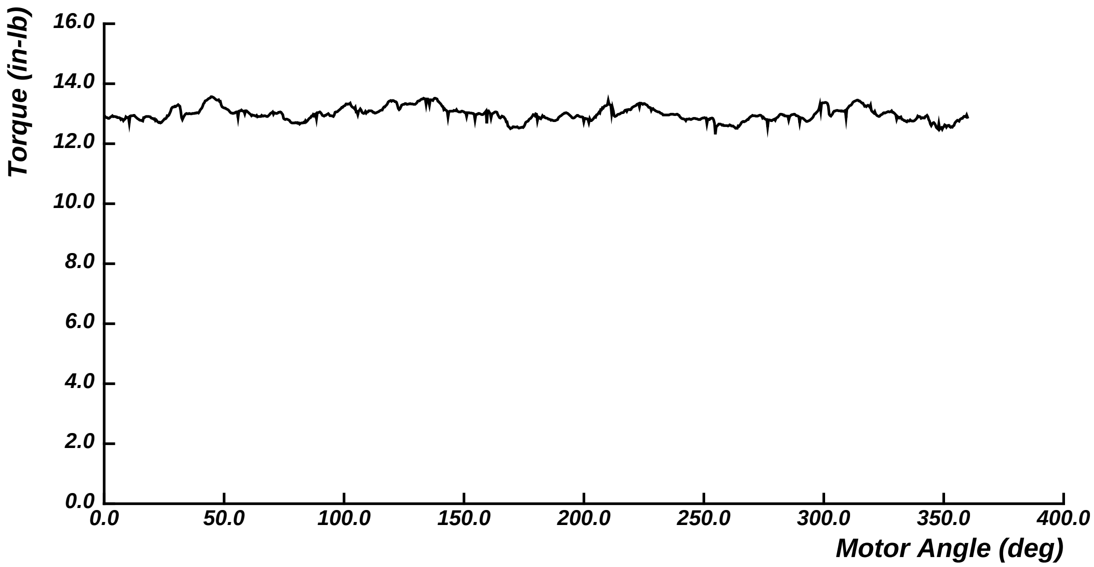

Figure 5.7: **Motor Torque versus Motor Angle for a command of 9 volts with feedforward compensation.** Figure 5.5 shows the ripple at 9 volts before compensation.

pensate for ripple, the controller multiplies the value of the ripple for the current motor angle with the absolute value of the commanded torque and adds this additional torque to the commanded torque.

For effective control, new ripple offset values need to be applied faster than the motor crosses ripple crests. For a maximum motor velocity of 7500 rpm, and a ripple frequency of 4 per/revolution, the torque ripple will have a frequency of 600 Hz. To be effective at motor velocities in this range, the servo would have to run 10 times faster, on the order of 6 kHz. There are two factors that argue against a rate of 6 kHz.

First, the field generated by the Samarium Cobalt magnets may be distorted by the magnetic fields induced in the stator pole pieces. This would cause a change in the phase angle of the ripple as the velocity of the motor increased. The magnitude of this effect is

Figure 5.8: **Motor Torque versus Motor Angle for a command of 0 volts with feedforward compensation.** Figure 5.4 shows the ripple at 0 volts before compensation.

unknown. Fortunately, at high motor velocities (and therefore high ripple frequencies) the motor and robot inertia filter the effects of the ripple. For the maximum motor velocity of 7500 rpm, the ripple frequency is 600 Hz. The last link of the manipulator has the highest natural frequency for the robot at close to 60 Hz with roll-off of 40db/decade after this frequency. Therefore, a ripple at 600 Hz is attenuated by 1/100 and can be effectively ignored.

The roll-off of the effect of ripple with motor velocity implies that only ripple frequencies on the order of the open-loop manipulator bandwidth are important. For an optimistic bandwidth of 60 Hz, ripple is only important for motor speeds below 750 rpm. A conservative estimate of the required ripple servo frequency is 10 times the ripple frequency or 600 Hz.

The application of the feedforward compensation reduces the amount of motor cogging by approximately 50 percent. Figures 5.7 and 5.8 show the motor torques for a 9 volt and 0 volt command after the application of torque ripple compensation. The graphs are for near zero motor speed.

Friction Compensation

The friction compensation $\hat{\mu}\, sgn\,(\tilde{q})$, derived in section 3.3.1, is a switching controller on the difference between the actual motor position and the desired position. The frequency of this switching function must be beyond the manipulator's bandwidth so that the switching is filtered by the actuator and manipulator dynamics for a required frequency of about 1 kHz.

The magnitude of the switching term is determined by measuring the motor voltage command required to make a joint move. Values around 0.3 volts were obtained for the all the motors (see page 5.2).

5.3.3 Servo Processor

Processor two runs the joint servo. This processor has the desired position, velocity, and acceleration terms resident in memory. It converts the motor positions and velocities, resident on processor 3, to joint coordinates. The resulting joint values are used to compute the feedforward gravity torques, the stiffnesses control torques, and damping torques. All the torques are added to give the total joint torque which is converted to motor command voltages and written to the output processor. The stiffness control torques are maintained as a separate value for use by the trajectory processor. The bandwidth on this process is 300 Hz.

Three processors are performing the real-time control functions. Two of the three are being used to condition the signals coming from and going to the motor controllers. The system could be improved, and two processors freed, if the input and output functions were performed by dedicated processors, one for each controller, that were not on the VME bus. This development is currently being considered.

Gravity Compensation

Compensation for gravity takes the form of feedforward torques to the last three motors based on the current joint angles. As is discussed in section 3.3.1, the torques due to gravity take the form:

$$G(q) = F(q)\,\hat{a}, \tag{5.1}$$

where $F(q)$ is a function of the kinematics and $\hat{a}$ is a set of parameters that are determined by the geometry of the loads.

The form of $F(q)$ can be determine from applying the Newton-Euler equations of motion symbolically to a model of the manipulator. Our model of the manipulator consists of the mass of the two manipulator links (M_3, M_4), and a load mass (M_l). We assumed that the center of mass of link 3 is at $[\,{}^PC_{3x} \quad 0 \quad {}^PC_{3z}\,]$, where the distances are described in frame three. The center of mass of link 4 is at $[\,{}^PC_{4x} \quad 0 \quad {}^PC_{4z}\,]$. Because the links are symmetric about their x-z plane, the mass centers have no y component. For simplicity in initial modeling, the load is assumed to be a point mass at a distance L_4 along the axis of joint 4.

The gravity equations which result from this geometric model and the MIT-WAM kinematics are given by:

$$F(q) = \begin{bmatrix} S_2S_4 - C_2C_3C_4 & C_2C_3 & -S_2 & -C_2C_3S_4 - C_4S_2 \\ C_4S_2S_3 & -S_2S_3 & 0 & S_2S_3S_4 \\ C_3S_2S_4 - C_2C_4 & 0 & 0 & -C_2S_4 - C_3C_4S_2 \end{bmatrix} \tag{5.2}$$

$$\tag{5.3}$$

$$\hat{a} = g \begin{bmatrix} M_lA_4 + M_4PC_{4x} - M_4PC_{4z} \\ M_4A_3 + M_lA_3 + M_3PC_{3z} \\ M_4L_3 + M_3L_3 + M_3PC_{3z} + M_lL_3 \\ M_lL_4 \end{bmatrix}. \tag{5.4}$$

By measuring the torques necessary to support the gravity load in one-hundred different locations and performing a regression fit to the data, an estimate of $\hat{a}$ of [2.2 3.3 108.8 6.1] in-lb was obtained.

5.3.4 Trajectory/Contact Processor

The trajectory processor and the main processor form the bulk of the control software. The trajectory processor's principal function is to calculate new desired joint positions, velocities, and accelerations, using the trajectory interpolation schemes described in section 4.4. In addition, for each trajectory servo cycle, the processor calculates the new stiffness gain matrix:

$$K_j = J^T K J, \quad \text{where } J \text{ is the manipulator Jacobian.}$$

For contact tasks, the processor determines if a contact has occurred during a servo cycle by monitoring the value of:

$$e = \tilde{q}^t K_j \tilde{q}, \quad \text{where } \tilde{q} = q_{desired} - q.$$

This value is the "potential energy stored" in the stiffness gains. When this value exceeds a threshold of $1\text{lb} - \text{in}^2$, a contact message is sent to the main processor. The maximum servo rate that could be achieved with all the computations was 150 Hz which is sufficient to generate smooth manipulator motions. Communication between the trajectory processor and the servo processor is maintained with pointers into the servo processors memory.

Additionally, the trajectory processor calculates all the interpolation splines for a list of desired joint motions, maintains a list of executable joint trajectories, and maintains the executable joint trajectory.

The trajectory code was originally part of the main code but, memory size constraints required us to move it to a separate processor. The implementation of the trajectory code makes the separation of the processors transparent to the user. Output from the trajectory program that would appear on the trajectory processor's virtual terminal has been coerced to appear on the main virtual terminal thus, making the trajectory program appear as if it were running on the main processor.

5.3.5 Main Processor

The main processor serves as the user interface to the entire system as well as the task control program. The interface has three menu levels. One menu controls the servo gains and feedback terms for the entire controller. The user can change any servo gain including gravity values, friction values, and filter gains, which makes experimentation easy.

Another menu controls data logging for latter plotting and analysis. The main processor can be placed in a servo loop that will record all of the system variables at a desired frequency for a specified duration. These values can be written to files or plotted over the bus-to-bus adapter on the workstation X window system.

The last menu controls trajectory generation. The menu gives options to teach joint motions, to read files of trajectories, and to move the arm interactively in joint, cartesian, and line space. In addition, the contact calculation, which gives the location and magnitude of contacts, can be performed on demand.

Lastly, at this level of abstraction, programs to perform WAM tasks can be written. Programs can be written to exploit any of the underlying features of the control system. Functions to perform compliant moves in all of the spaces have been written. Moves can be terminated upon contact, or upon the loss of contact. The contact calculation can be applied to calculate new stiffness and line frames which can be applied to compliant moves. These functions can be combined to form a high level task planner, which is one future area of research.

Chapter 6

Results

The principal goal of this research was to create a control system and software structure to demonstrate the feasibility of whole-arm manipulation with the MIT-WAM robot. Toward this end, we have implemented the compliant control system described in the previous chapters, in conjunction with the manipulator kinematics and trajectory generation, to create a simple task programming environment. Within this environment, we have written a program that uses the manipulator to search for three boxes in the workspace, and to move compliantly over each of these boxes. This demonstration requires open-loop force commands, compliant motions until contact and while in contact, and line motions. The ability of the robot to determine contact locations and forces was also tested.

In addition, we have determined the performance of the manipulator in a few key areas. The stiffness of the transmission was measured to determine the amount of positioning error that could occur through compliance. The stiffness of the transmission also limits the maximum controllable stiffness of the manipulator. We briefly examined the dynamic responses of the first and last links. These measurements were useful for determining the necessary control servo rates. We examined the trajectory repeatibility for the endtip with and without integral control. Lastly, the ability of the robot to accurately apply a desired force along the last link was measured in a number of configurations.

6.1 Static Performance Tests of Manipulator

6.1.1 Linkage Stiffness

The stiffness of the transmission generally limits the stiffness that can be achieved by position feedback. For the MIT-WAM manipulator the stiffness of the transmission is controlled by the free length of cable and is lowest for the fourth joint. The stiffness was measured by locking the robot's joints and then applying torques to the motor shaft. The resulting motion of the motor was recorded using the motor resolver.

The maximum torque that could be applied to the shaft was 12 in-lb. At this value two cable strands snapped. Up to this torque value, the rotation of the motor shaft corresponded to a stiffness of 40.3 in-lb/rad. The motor stiffness is increased by the transmission ratio squared at the joint to 16,000 in-lb/rad. This is the highest stiffness (as seen from the fourth joint) that the controller could achieve simply with position feedback. We have achieved values of 1000 in-lb/rad at this joint. Instability at higher gain values is probably due to the phase lag in the filters introduced into the servo controller and the phase lag in the R/D converter. The base transmission is significantly stiffer than the last joint (because of the shorter free length of cable) and controlled stiffnesses up to 5000 in-lb/rad have been achieved.

6.1.2 Bandwidth

The natural frequencies of the robot are important in determining the control bandwidth that can be achieved without dynamic modeling. The natural frequencies are position dependent because of the changing inertia of the robot. Bounds on the frequencies can be found by examining the frequency for the last joint, which has the highest frequency, and the frequency of the first joint which has the lowest frequency.

The frequencies of the last joint depend on the stiffnesses at the second and third joints which is controlled by the servo gains at these joints. In order to bound the maximum, we measured the natural frequency with the end of link three affixed to the floor with a steel beam. Square wave commands were given to the motor at varying frequencies and the velocity of the motor was measured using the velocity output of the R/D converter. Both the input and velocity output were measured with an HP spectrum analyzer. The resulting bode plot indicated that the transfer function from motor commands to motor velocity had a complex pair of poles at 56 Hz (the natural frequency) and a complex pair of zeros at 32 Hz. This is the frequency of the interaction between the motor and the link mass through the transmission stiffness. The damping ratio on both the poles and zeros is about 0.3.

The natural frequency of the base joint was measured in several configurations. The lowest frequency occurs with the arm horizontal and full extended. In this configuration, a complex pair of poles was observed at 44 Hz. The damping ratio on the poles is near 0.6 for a very well damped system.

The control system should be able to achieve closed-loop bandwidths of 5 Hz without the phase lag from the manipulator dynamics effecting the controller. By measuring the response of the manipulator in the horizontal configuration to a step disturbance in θ_1, we found a closed-loop natural frequency of 2 Hz and a damping ratio of 0.5 for position gains of (4000, 3500, 300, 500) in-lb/rad in each of the joints.

6.1.3 Range and Accuracy of Open-Loop Forces

Our approach to force control is to apply forces open-loop at desired locations. The ability of the manipulator to apply a desired force was measured in two configurations at a single point along the link. The manipulator was placed in a fully extended horizontal configuration using joint stiffnesses, and then a commanded force was multiplied by the endpoint Jacobian to determine a feedforward torque command. A force sensor with a range of 0 to 28 lb and a resolution of 0.25 lb was located at the endpoint to measure the effect of the additional torque.

In both directions ${}^5\hat{x}$ and ${}^5\hat{y}$, the measured output force correlates closely with a linear function of the commanded torque. The ratio of the commanded force to actual force in the ${}^5\hat{x}$ direction, in this configuration, is 1.03 and in the ${}^5\hat{y}$ direction 1.16. Neither graph passes exactly through the origin. The ${}^5\hat{x}$ graph has a commanded force intercept of 0.25 lb and the ${}^5\hat{y}$ graph has an intercept of 0.5 lb.

There are a number of possible factors contributing to the deviations from the desired relationship. First, there is some vibration in the robot which causes the force measurement to fluctuate by ± 0.5 lb. The vibration is probably caused by limit cycles in the control loop through the friction in the system. The torque ripple remaining after the application of feedforward compensation can cause variations in the force magnitude of

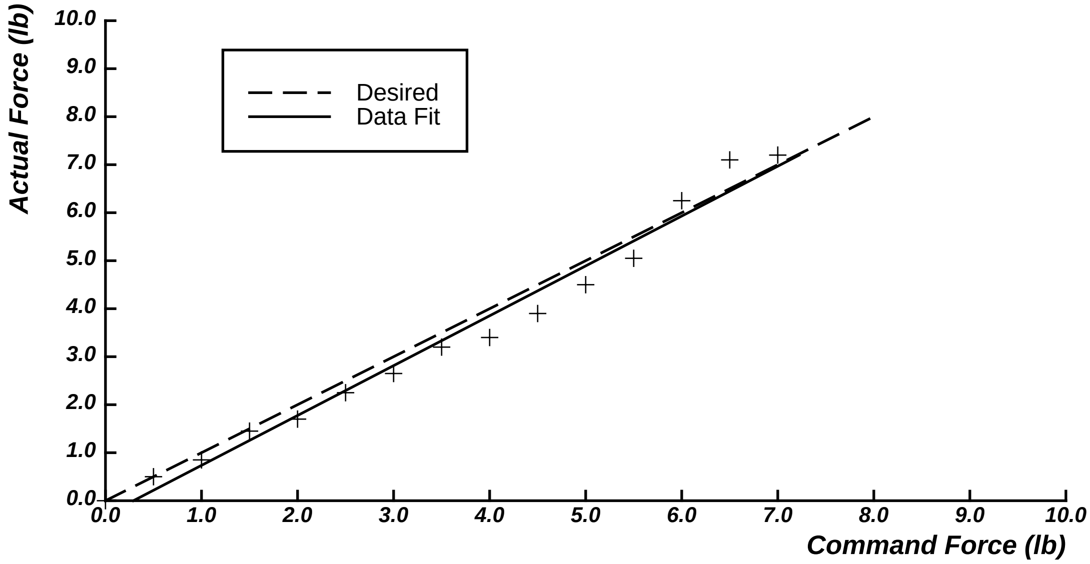

Figure 6.1: **Actual Force versus Commanded Force in the $\hat{x}_5$ direction.** Force was applied at the endpoint with joint angles of (0, $-\pi/2$, 0, 0). Commanded forces greater than 7.0 lb cause a loss of pretension in joint 4.

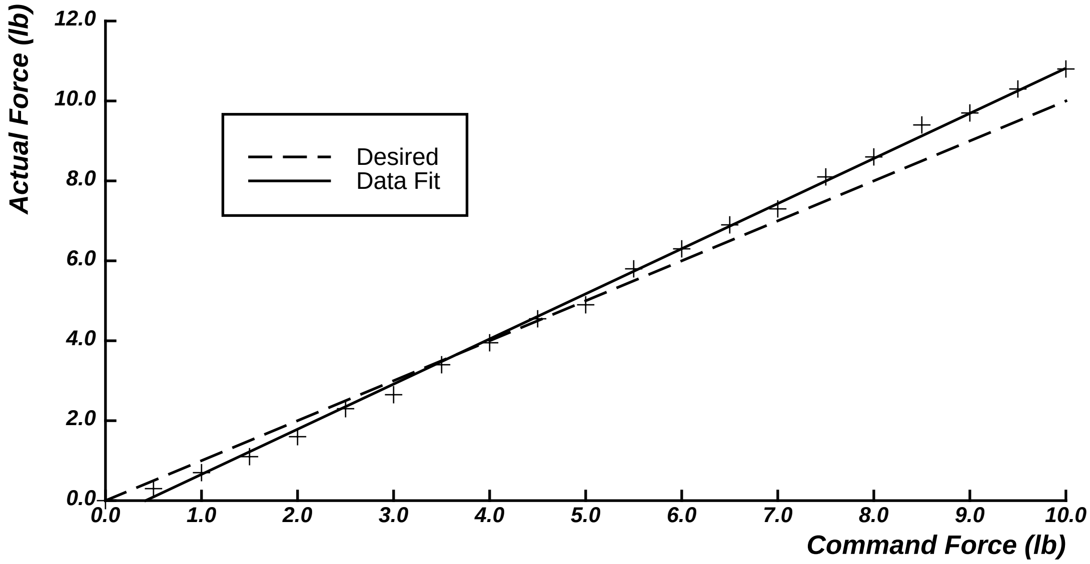

Figure 6.2: **Actual Force versus Commanded Force in the $\hat{y}_5$ direction.** Force was applied at the endpoint with joint angles of (0, $-\pi/2$, 0, 0). Commanded forces greater than 10.0 lb caused a loss of pretension in the differential drive.

about ± 0.2 lb. Lastly, the motor torque-voltage curve could be in error by a few percent from calibration errors or thermal drift in the motor torque constant.

The range of forces that the manipulator can apply is direction and configuration dependent. In the location used for the test, the manipulator can exert a theoretical maximum of 18 lb of force in the vertical direction at the end-effector. This value is limited by the torque output of the last motor. The loss of pretension in the last joint limits the effective range to 7.0 lb in the vertical direction. The pretension is limited by the yield strength of the cable. Pretension is lost in one of the differential drive motor cables at a value of 10.0 lb in the $\hat{y}_5$ direction.

6.1.4 Repeatibility

The repeatibility of the manipulator was measured by determining the maximum deviation that occurred when the manipulator was successively commanded to a desired position. For joint angles of (0.0, -1.57, 0.0, 0.0) and stiffness control the endpoint deviated by $\pm 1/2$ in. With integral control, the deviation was $\pm 1/32$ in. There was not any increase in the error with each successive motion from zero to this position. The repeatibility is a primarily a function of the joint gains, the friction, and deadzone.

6.2 Whole-Arm Tasks

We have experimented with two whole-arm tasks. We examined the effectiveness of the robot as a contact sensor. As was shown in section 4.2.3, the ability of the robot to determine contact locations and forces is position and configuration dependent. We have measured the variation of contact determination as the point of contact application moves with constant force and with increasing force levels at a single contact location.

The second task is a demonstration of searching to show the ability of the robot to perform compliant motions along the entire last link. The robot contacts three boxes, noting the angles at which contact occurred, and then compliantly moves up, over, and down each box. The manipulation demonstrates open-loop force commands, compliance control, contact detection, and line motions. The search procedure is implemented as a set of higher level primitives.

6.2.1 Contact Location Identification

The contact problem is highly position dependent; therefore, we have tested the ability of the manipulator to determine contact forces and locations in two configurations that are often used. The first is the horizontal full extended configurations (joint angles [0, $-\pi/2$, 0, 0]). In this configuration we applied a set force of 2.75 lb in the ${}^5\hat{x}$ and ${}^5\hat{y}$ directions (see figure 4.2) to the manipulator at varying distances along the second link. Figures 6.3 and 6.4 shows the variation in the estimate of the contact force and contact location as the contact location varies. The contact force estimate is near 1.5 lb for all contact locations which is approximately 1/2 the desired value of 2.75 lb. The contact location estimate is generally twice the actual contact location and varies greatly as the contact location changes. The sensitivity of the contact location calculation in this configuration can be seen by noting how a small deviation in the force value from the fitted line leads to a large deviation in the location estimate from its fitted line.

The cause of the factor of approximately two evident in both the force and location

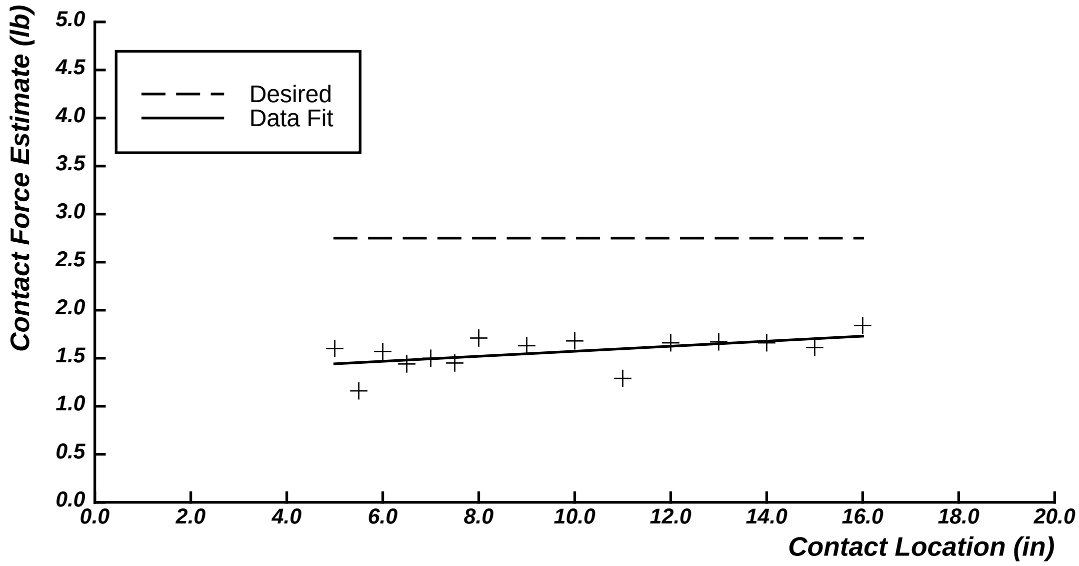

Figure 6.3: **Contact Force Estimate versus Contact Location - X force.**

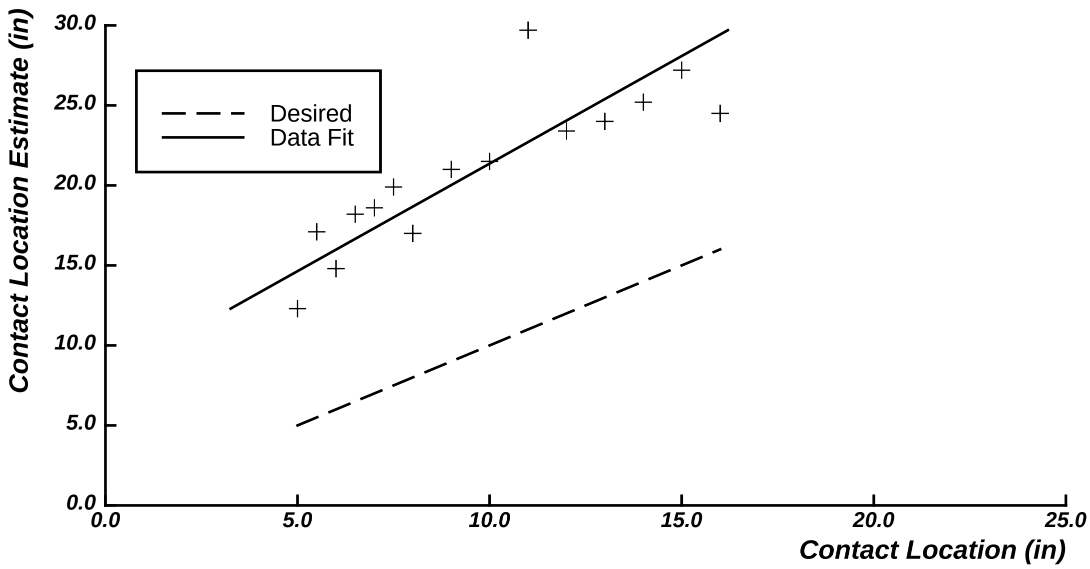

Figure 6.4: **Contact Position Estimate versus Contact Location - X force.** In both figures a force of 2.75 lb was applied in the $\hat{x}_5$ direction with manipulator angles of $(0, -\pi/2, 0, 0)$.

estimates is unclear. One possible cause is the combination of the effects of gravity and friction. In the tested configuration it is possible to move the end of the manipulator by $\pm 1/2$ in without having it return to the desired location. This can cause an error in the estimate of the torque about the last joint, to which the calculation is very sensitive. Additionally, we have found that the results depend on the stiffness gains. Lastly, in this configuration the solution to the contact problem diverged for around 10% of the attempted measurements. We conclude that the contact problem in this configuration is not well conditioned. Figures 6.5 and 6.6 show the same sensitivity in the contact calculation for a changing applied force in the ${}^5\hat{x}$ direction at the end of the manipulator.

WAM tasks also involve forces in the ${}^5\hat{y}$ direction. In the previous configuration, a ${}^5\hat{y}$ contact force cannot be resolved; therefore, we tested this direction in the configuration [0, $-\pi/2$, 0, -1]. Figures 6.7 and 6.8 show the variation in the estimate with the contact location for force in the $\hat{y}_5$ direction. Similarly, figures 6.9 and 6.10 show the variation in the estimates with contact force in the ${}^5\hat{y}$ direction applied at the endpoint.

In this configuration, the manipulator is able to determine contact forces and locations with accuracy of approximately 30% which is considerably better then in the previous test. Again, the source of the error is unclear. Some error maybe due to the accuracy of the forward problem (examined in the previous section) of applying desired forces at the endtip. In either direction, we found that the contact is very configuration and gain dependent, and is sensitive to errors in calibration and nonlinearities in the system. A better theory of the sensitivity of the calculation to these parameters is a topic of future research.

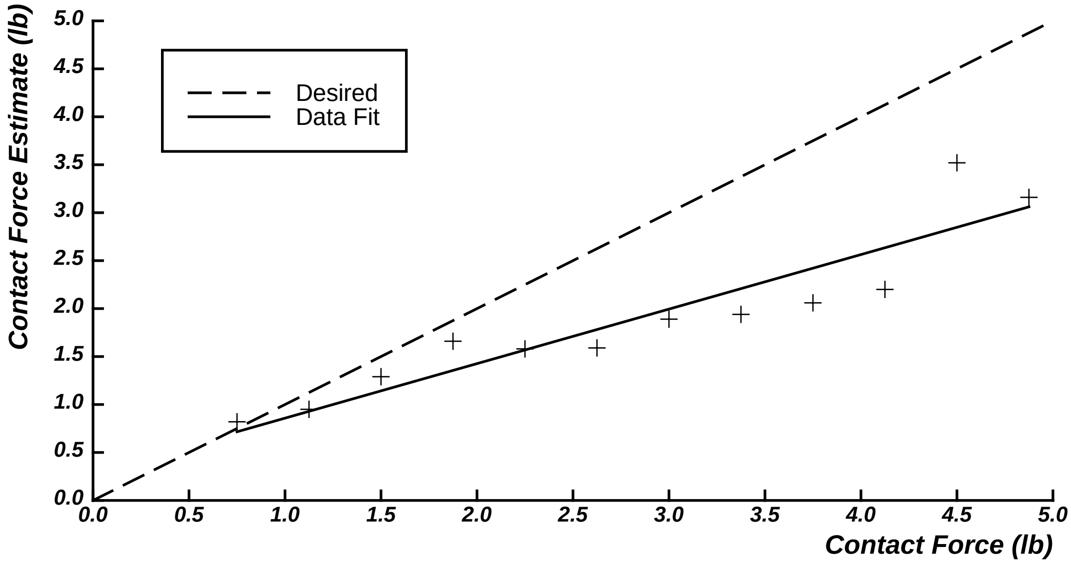

Figure 6.5: **Contact Force Estimate versus Contact Force - X force.**

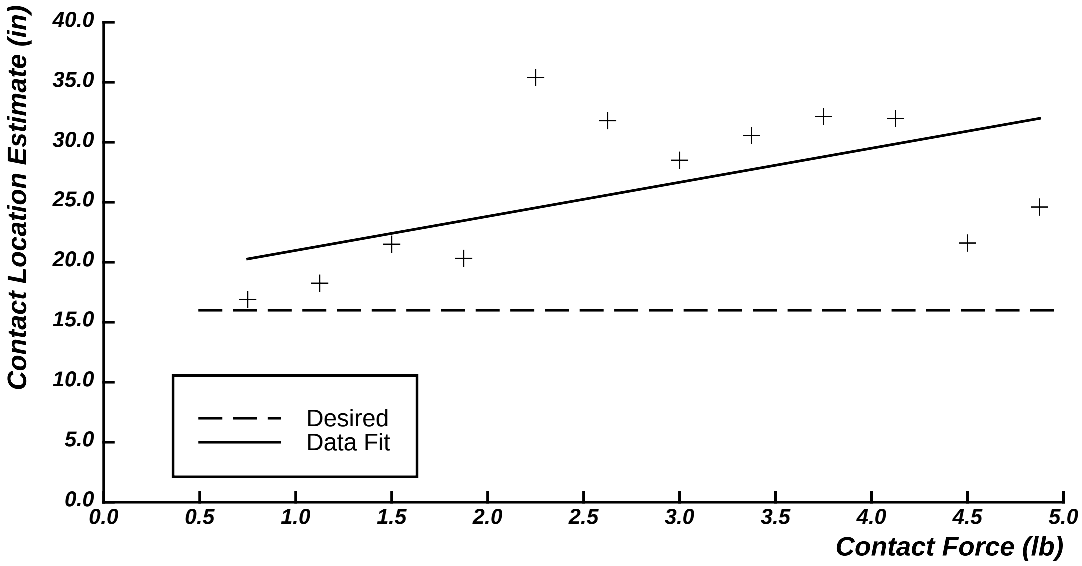

Figure 6.6: **Contact Position Estimate versus Contact Force - X force.** In both figures a varying force was applied in the $\hat{x}_5$ direction at the endpoint of the manipulator, 16.0 in, with joint angles of $(0, -\pi/2, 0, 0)$.

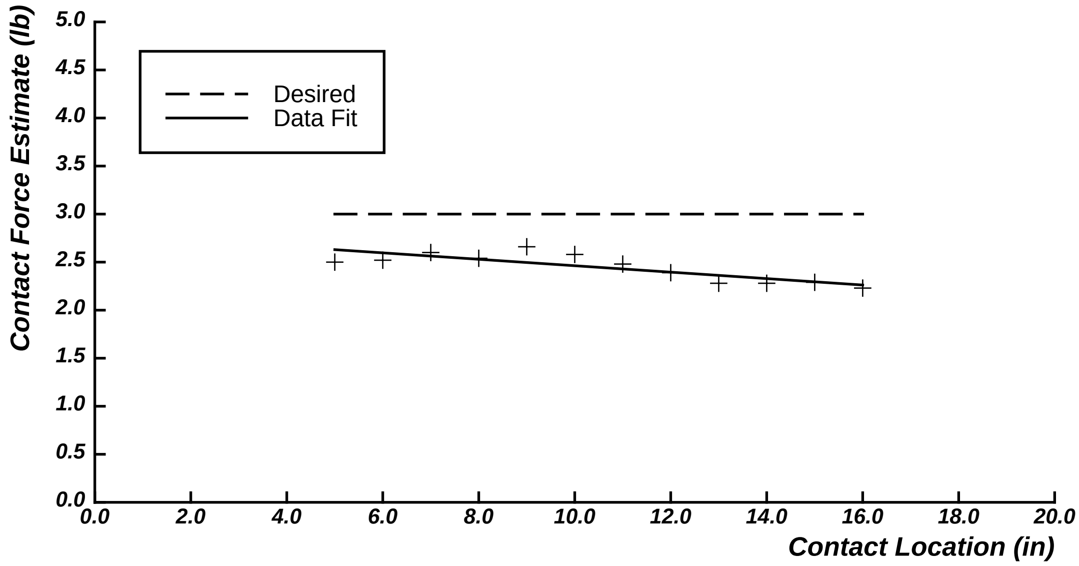

Figure 6.7: **Contact Force Estimate versus Contact Location - Y force.**

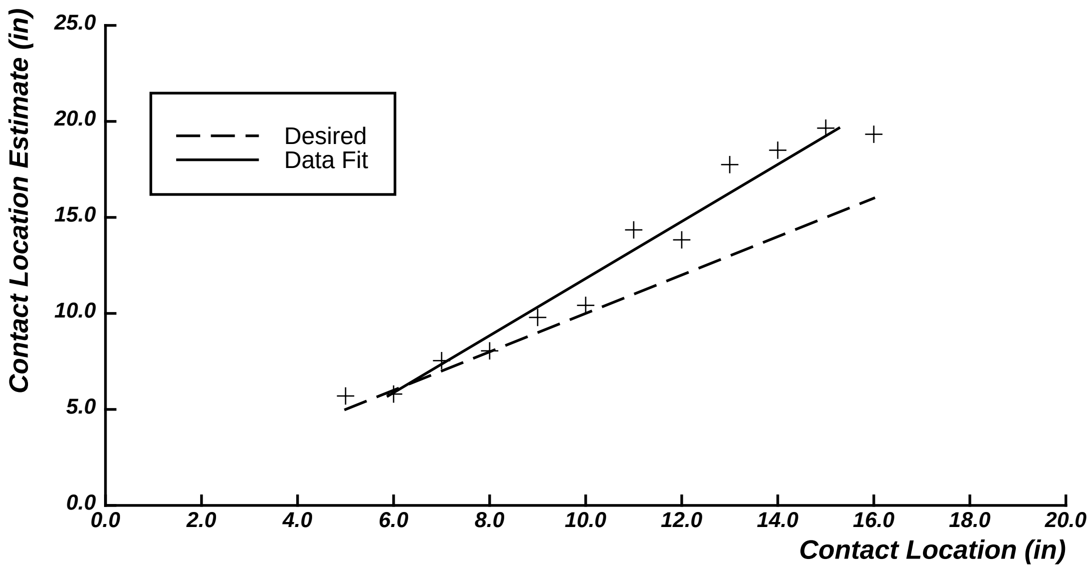

Figure 6.8: **Contact Position Estimate versus Contact Location - Y force.** In both figures a force of 3 lb was applied in the $\hat{y}_5$ direction with manipulator angles of (0, $-\pi/2$, 0, -1).

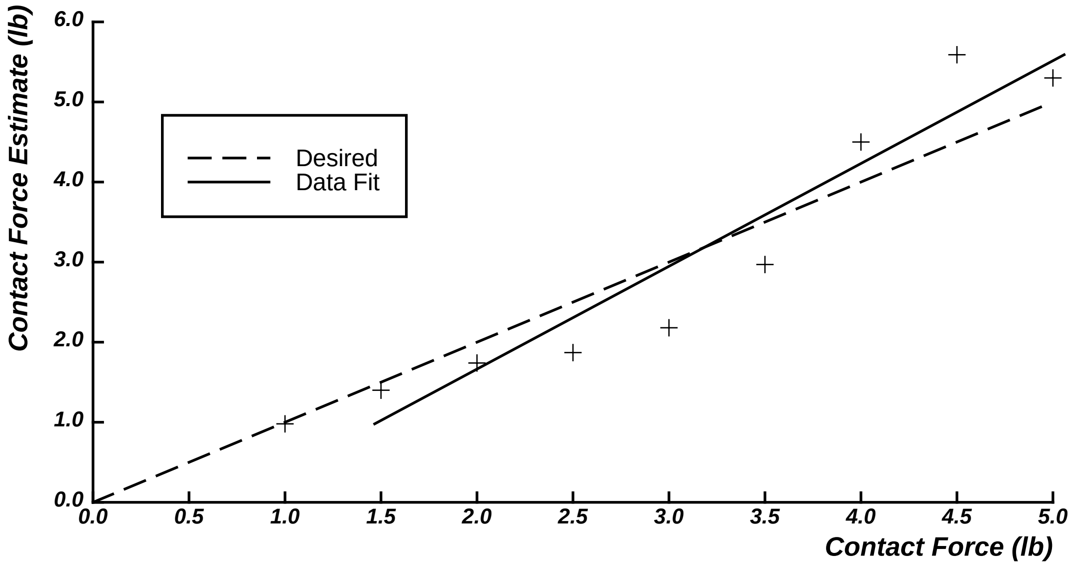

Figure 6.9: **Contact Force Estimate versus Contact Force - Y force.**

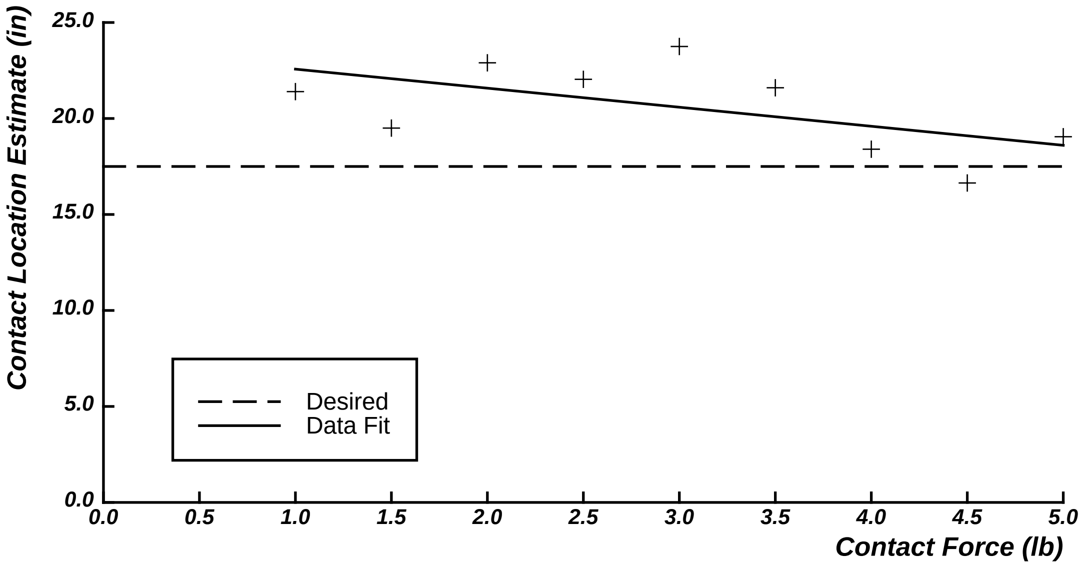

Figure 6.10: **Contact Position Estimate versus Contact Force - Y force.** In both figures a varying force was applied in the $\hat{y}_5$ direction at the endpoint of the manipulator, 17.5 in, with joint angles of $(0, -\pi/2, 0, \text{-}1)$.

Lastly, preliminary results show that the manipulator cannot distinguish forces and contacts in the $\hat{z}_5$ direction; however, this direction is seldom used. Except for collisions with objects at the endpoint, manipulations involve manipulating objects with the sides of the links and involve primarily forces in the $\hat{x}_5$ and $\hat{y}_5$ directions.

6.2.2 Searching and Pushing a Set of Boxes

We have demonstrated some of the aspects of whole-arm manipulation with a searching task. Three boxes, of varying sizes, are arranged on a table within the workspace of the last link. The task is to move the manipulator through the workspace so as to locate the edges and determine the sizes of each of the boxes. The control program is given the knowledge that the shapes are rectangular and that there are three objects.

To perform the search, the manipulator begins by moving down, checking for collision, to the level of the table, which is at the height of the base joint. The manipulator then rotates about the base joint until contact occurs. This contact indicates the location of the first box. At this point the code for searching up, over, and down a box is invoked.

The search function, first applies a force in the direction of the box and decreases the position gain about joint 1. The robot then moves up using a line motion, so that the last link will stay horizontal, until contact is lost. At this point the gain for joint 1 is returned to the previous value and the manipulator is moved over a small amount in the direction of the applied force. This small motion makes the subsequent downward motion more robust.

The manipulator next moves down until contact is again made. The gains in joints 2 and 4 are decreased to decrease the effective stiffness of the manipulator in the direction of the box. The robot is then commanded to move into the box in order to create a bias force. The manipulator then rotates about joint 1 until contact is again lost. This defines the width of the box. A final downward line motion completes the compliant search.

To move to the next box, the manipulator is realigned with the horizontal plane and then rotated about the base joint until contact occurs with the next box. The box tracing function is then called again and the procedure repeats.

Some of the code for implementing this procedure is sketched below:

```
search_demo() {
        if((ret = move_abs_until_contact(2.0, 0.0, -1.65,
        0.0, 0.0)) == CONTACT)
                print(Premature contact! Aborting)
                goto error
                }

        if((ret = move_until_contact(8.0, 2.0, 0.0,
        0.0, 0.0)) == END_TRAJECTORY){
                print(No Objects in the robot's path! Aborting!)
                goto error
                }

        record_positions(object_location[0])
        apply_bias_force(dir,1.5)

        if((ret = trace_object(dir,object_location[0])) ==
        END_TRAJECTORY){
                print(Error in trace object! Aborting!)
                goto error
                }

        move_to_until_done(1.0, (current_position[0]+0.05),
        -1.65, 0.0, 0.0)
        print(object 1)
```

```
        ⋮
Repeat for each box }


trace_object(dir,angles) {
Search Up
        array_set(KP,0,0,600.0)
        return_on_initialization()
        if((ret = move_line_by_while_contact(Z_hat, 20.0,
        5.0, DEFAULT_TOL,NULL)) == END_TRAJECTORY)
                goto error

        move_to_until_done(0.5, current_position[0],
        current_position[1], current_position[2], current_position[3])
        force_zero()

Search Across
        array_set(KP,0,0,2000.0)
        array_set(KP,1,1,800.0)
        array_set(KP,3,3,200.0)

        move_by_until_done(0.5, 0.05, 0.0, 0.0, 0.0)
        if(( ret = move_line_by_until_contact(Z_hat, -8.0, 2.0
        DEFAULT_TOL, NULL)) == END_TRAJECTORY)
                goto error

        move_by_until_done(0.5, 0.0, -0.1, 0.0, -0.1)
        if((ret = move_while_contact(5.0, 1.0, 0.0, 0.0, 0.0))
        == END_TRAJECTORY)
                goto error

Search Down
        ⋮
}
```

This procedure is meant only to demonstrate the feasibility of whole-arm manipulation applied to searching. The idea can be greatly extended by including geometric modeling and identification of the environment. Currently, all the robot records is the joint angles at the initial contact. Far more information could be obtained by recording the entire path of the manipulator and using this information to make determinations of the objects shape and some of its properties. Additionally, simple models of the expected objects could be provided and the manipulator could plan compliant paths that would locate and follow the actual shapes of the object.

Lastly, the current scheme does not update the contact force as a contact motion proceeds; therefore the manipulator will lose contact with an object with rapidly varying shape. The inclusion of the contact forces into the motion plan can be obtained by merging in the contact location procedure to update the manipulator location. Before this can be done, the reliability and speed of the contact calculation must be improved.

6.3 Conclusion

In this chapter, we have examined the performance of the manipulator and control scheme in a few areas. The repeatibility of the manipulator under stiffness control is not high by industrial standards. This can be improved by increasing the maximum stiffness gains of the robot or by lag-lead feedback. The stiffness gains do not seem to be limited by the stiffness of the transmission, rather phase lags introduced by the filtering in the compensation, computational time delays, and a delay in the R/D converter seem to be the limiting factors. The filters in the controller could be removed if the noise in the position signal could be eliminated.

We found that the manipulator can cleanly apply desired forces to the environment with a fair degree of accuracy. Errors in this task can be attributed to the remaining ripple, inaccurate knowledge of the motor torque constant, and friction. Improvements in the actuator would increase the performance significantly.

The inverse problem of determing the contact forces and locations given the joint torques does not work as well as anticipated. Intrinsic errors in the system maybe accentuated by the ill conditioning of the solution. A better theoretical model of the sensitivity of the results to errors in torque calibration, the gravity model, friction, transmission stiffness, and gains is needed before methods can be suggested to improve the technique.

Lastly, we have demonstrated some high-level procedures to perform a search using whole-arm motions. The procedure uses compliant motions and contact detection to find a set of boxes. The procedure is fairly robust to the location of the boxes on the surface. This procedure could be extended by including geometric modeling and higher level planning to make the system more robust and to extract more information from the system. Task level planning for whole-arm tasks to take advantage of the inherent robustness of procedures that use lines to effect tasks is a major area of future research.

Chapter 7

Conclusion

7.1 Review

We are interested in developing robots that can work in unstructured environments. Intentionally or not, a robot in such an environment will interact with its workspace along all of its moving surfaces. We therefore suggest that robots used in unstructured environments should be designed such that they can use all of their moving surfaces for manipulation, or for whole-arm manipulation (WAM). In this thesis, the kinematic and control issues involved in WAM have been explored and have been applied to the MIT-WAM.

WAM task can be grouped into pushing, searching, enclosure, and exclusion. These division form a basis for thinking about the range of tasks that can be accomplished with a whole-arm manipulator, and for developing planning algorithms to perform the tasks. Theoretically, the utility of a manipulator can be extended by using it as a sensor to detect contacts and the location and force of the contact. We show that in most configurations the joint torques are sufficient to determine the forces of contact and the contact location.

Manipulation in an unstructured environment requires the development of robust force control schemes to control the forces developed along all the surfaces of the manipulator. Robustness can be achieved by using the motor torques/currents to control the contact forces. In exchange for increased robustness, we lose the ability to modify the effective manipulator inertia and to decrease the ripple and other nonlinearities through feedback. Instead, we must rely on the performance inherent in the mechanical and electrical design of the manipulator and on feedforward compensation for nonlinear effects that can be effectively modelled.

In chapter 5, we examined the implementation and effectiveness of the feedforward compensation. By building a feedforward table to compensate for motor ripple and cogging, we were able to decrease the effective ripple by 50%. In the implementation, we found that the noise induced by PWM amplifiers was a major performance limitation.

Line motions provide a useful way of describing the kinematics of whole-arm tasks. To effect line motions for the MIT-WAM, we have shown how the kinematics of this robot can be solved to place the last link along a desired line. Line kinematics was then applied to a maximum deviation trajectory planning algorithm to produce line motions. We applied these motions, along with compliance control, to a search task to demonstrate the ability of the MIT-WAM to perform tasks that use the entire surface of a link.

Lastly, we found that the MIT-WAM robot is able to accurately apply a desired force, and that variations from the desired force arise primarily from nonlinearities in the motors and controllers. We found that the inverse problem of determining the forces and locations of contact form the motor torques is not well conditioned in many useful configurations, and that the same variations in the motors and controllers and gravity loads can create large variations in the sensed contact.

7.2 Future Work

To develop a whole-arm manipulator a number of areas will need further exploration. We have based our control design on the inherent performance of the manipulator and its actuators. The development of new high-performance actuators with less ripple, deadzone, and cogging and more torque would significantly improve the performance of our system. On the computational side, the Condor system is limited to eight processors of which we are currently using five. The effectiveness of the system would be enhanced by the development of more intelligent motor controllers to take over some of the computation.

The taxonomy of whole-arm tasks described in chapter 2 needs to be applied to the development of planning algorithms to automatically generate task level programs. This will also involve programming issues involved in the creation of a language to describe whole-arm tasks. The number of functions to implement the task level programs that we have developed seems to be increasing exponentially; therefore, a methodology for decreasing the complexity and organizing the programming task must be developed.

The kinematics of line manipulations and whole-arm tasks maybe extensible to more degrees-of-freedom and more robots acting in concert. A model of the sensitivity of the contact problem to the configuration of the robot and to disturbances needs to be developed. An understanding of this problem might lead to different kinematic configurations that are less sensitive to disturbances, or to the conclusion that force sensing is necessary along all of the links for contact determination.

7.3 Conclusion

We have examined the problem of whole-arm manipulation in a number of different areas in order to build-up a useful system for the control and experimentation of whole-arm tasks on the MIT-WAM. We have suggested a taxonomy of tasks to classify the various whole-arm tasks and suggested different types of manipulation. The problem of using the manipulator as a sensor was examined. Kinematic relations for the MIT-WAM that can be used to perform line manipulations were described. Issues involved with force control and mechanical design were reviewed to determine a robust force controller. Lastly, the developed system was applied to a whole-arm task to show the potential for this type of manipulation. Future work will concentrate on the development of better actuation systems and on the planning/programming issues involved with whole-arm manipulations.

Bibliography

[An 86] An, C.H., "Trajectory and Force Control of a Direct Drive Arm," MIT AI Technical Report AI-TR-912, MIT Artificial Intelligence Lab, Cambridge, MA, 1986.

[Asada, Kanade 81] Asada, H., and Kanade, T., "Design of Direct-Drive Mechanical Arms," *Journal of Vibrations, Acoustics, Stress, and Reliability, Transactions of the ASME,* vol. 105, no. 3, July 1983, pp. 312–316.

[Asada and Slotine 86] Asada, H., Slotine, J.E., *Robot Analysis and Control*, Wiley-Interscience, New York, New York, 1986.

[Asada 84] Asada, H., Youcef-Toumi, K., and Lim, S.K., "Joint Torque Measurement of a Direct-Drive Arm," *23d IEEE Conference on Decision and Control,* December 1984, pp. 1332–1337.

[Atkenson 86] Atkenson, C. G., "Roles of Knowledge in Motor Learning," MIT AI Technical Report AI-TR-942, MIT Artificial Intelligence Lab, Cambridge, MA, 1986.

[Cannon 83] Cannon, R. H. Jr., Rosenthal, D. E., "Experiments in Control of Flexible Structures with Non-colocated Sensors and Actuators," *AIAA Journal of Guidance and Control*, September-October 1984, vol. 7, no 5., pp. 546-553.

[Craig 87] Craig, J.J., *Introduction to Robotics Mechanics & Control*, Addison Wesley, Reading, MA, 1987.

[Chiu 85] Chiu, S.L., "Generating Compliant Motion of Objects with an Articulated Hand," S.M. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, June 1985.

[DiPietro 88] DiPietro, D.M., "Development of an Actively Compliant Underwater Manipulator," S.M. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, May 1988.

[Eppinger 87] Eppinger, S.D., and Seering, W.P., "Understanding Bandwidth Limitations in Robot Force Control," *proc. 1987 IEEE International Conference on Robotics and Automation,* Raleigh, N.C., April 1987.

[Hogan 84] Hogan, N., "Impedance Control: An Approach to Manipulation," *ASME Journal of Dynamic Systems, Measurement, and Control*, March 1985, vol. 107, pp1-24.

[Jacobsen 84] Jacobsen, S.C., Wood, J.E., Knutti, D.F., Biggers, K.B., "The Utah/MIT Dexterous Hand: Work in Progress," *International Journal of Robotics Research*, vol. 3, no. 4, 1984.

[Jacobsen 86] Jacobsen, S.C., Iversen, E.K., Knutti, D.F., Johnson, R.T., and Biggers, K.B., "Design of the Utah/MIT Dexterous Hand," *proc. 1986 IEEE International Conference on Robotics and Automation,* San Francisco, CA, April 1986.

[Kazerooni 85] Kazerooni, H., "A Robust Design Method for Impedance Control of Constrained Dynamic Systems," Ph.D. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, February 1985.

[Lawrence 88] Lawrence, D.A., "Impedance Control Stability Properties in Common Implementations," *proc. 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988.

[Lehtomaki 81] Lehtomaki, N.A., "Practical Robustness Measures in Multivariable Control System Analysis," Ph.D. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, May 1981.

[Luh 83] Luh, J.Y.S., Fisher, W.D., and Paul, R.P., "Joint Torque Control by a Direct Feedback for Industrial Robots," *IEEE Transactions on Automatic Control,* AC-28, 1983, pp. 153–160.

[Mason 81] Mason, M.T., "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11, June 1981, pp. 418-432.

[Mason and Salisbury 85] Mason, M.T., Salisbury, J.K., *Robot Hands and the Mechanics of Manipulation*, MIT Press, Cambridge, MA, 1985

[Narasimhan 87] Narasimhan, S., "Dexterous Robotic Hands: Kinematics and Control," S.M. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, October 1987.

[Narasimhan 88] Narasimhan, S., Siegel, D.M., Hollerbach, J.M., "Condor: A Revised Architecture for Controlling the UTAH-MIT hand," " *proc. 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988.

[Nayef and Mook 79] Nayef, A.H., Mook, D.T., *Nonlinear Oscillations*, John Wiley & Sons, New York, NY, 1979

[Nevins and Whitney 73] Nevins, J.L., Whitney, D.E., "The Force Vector Assembler Concept," *proc. of the First CISM-IFToMM Symposium,* Udine, Italy, September 5–8, 1973, pp. 273–288.

[Paul 87] Paul, B.J., "A Systems Approach to the Torque Control of a Permanent Magnet Brushless Motor," S.M. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, August 1987.

[Raibert and Craig 81] Raibert, M.H., and Craig, J.J. "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems, Measurement, and Control,* ASME, vol. 103, no. 2, June 1981, pp. 126–133.

[Rosenberg and Karnopp 83] Rosenberg, R.C., Karnopp, D.C., *Introduction to Physical System Dynamics*, McGraw-Hill, New York NY.

[Salisbury 80] Salisbury, J.K., "Active Stiffness Control of a Manipulator in Cartesian Coordinates," Proc. of 19th Conference on Decision and Control, IEEE, vol. 1, December 1980.

[Salisbury 84a] Salisbury, J.K., "Interpretation of Contact Geometries from Force Measurements," *proc. 1st International Symposium on Robotics Research,* Bretton Woods, NH, September 1984b, published by the MIT Press, Cambridge MA.

[Salisbury 84b] Salisbury, J.K., "Design and Control of an Articulated Hand," *International Symposium on Design and Synthesis,* Tokyo, Japan, July 1984a.

[Salisbury 87] Salisbury, J.K. "Whole-Arm Manipulation," Proceedings of the 4th International Symposium of Robotics Research, Santa Cruz, CA, August 1987. Published by the MIT Press, Cambridge MA.

[Salisbury 88] Salisbury, J.K., Townsend, W.T., Eberman, B.S., DiPietro, D., "Preliminary Design of a Whole-Arm Manipulation System (WAMS)," *proc. 1988 IEEE International Conference on Robotics and Automation,* Philadelphia, PA, April 1988.

[Shimano 78] Shimano, B.E., "The Kinematic Design and Force Control of Computer Controlled Manipulators," Ph.D. thesis, Department of Computer Science, Stanford University, March 1978.

[Strang 86] Strang. G., *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Cambridge MA.

[Taylor 79] Taylor, R.H., "Planning and Execution of Straight Line Manipulator Trajectories," IBM Journal of Research and Development, Number 23, 1979.

[Townsend 88] Townsend, W.T., "The Effect of Transmission Design on Force-Controlled Manipulator Performance," Ph.D.

thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, April 1988.

[Wlassich 86] Wlassich, J.J., "Nonlinear Force Feedback Impedance Control," S.M. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, February 1986.

[West 87] West, H., "Kinematic Analysis for the Design and Control of Braced Manipulators," Ph.D. thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, June 1987.

[Whitney 77] Whitney, D. E., Whitney, D.E. "Force Feedback Control of Fine Manipulator Motions," *Journal of Dynamic Systems, Measurement, and Control,* vol. 99, no. 2, June 1977, pp. 91–97.

[Whitney 85] Whitney, D.E., "Historical Perspective and State of the Art in Robot Force Control," *proc. 1985 IEEE International Conference on Robotics and Automation,* St. Louis, MO, March 1985, pp. 883–889.

[Wu 80] Wu, C.H., and Paul, R.P., "Manipulator Compliance Based on Joint Torque Control," *19th IEEE Conference on Decision Control,* December 1980, pp. 88–94.

[Youcef-Toumi 85] Youcef-Toumi, K., "Analysis, Design and Control of Direct-Drive Manipulators," Ph.D thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, May 1985.

Appendix A

Effects of Friction Compensation

Consider a dynamic system of the form:

$$\ddot{x} + x + \hat{\mu}\, sgn\,(x) + \mu\, sgn\,(\dot{x}) = 0. \tag{A.1}$$

As was shown earlier in section 3.3.1, this equation is stable for $\hat{\mu} \leq \mu$ and has critical points only in the neighborhood of the origin. This appendix will demonstrate through the use of averaging that for any small positive μ and $\hat{\mu}$ this equation is stable. A development of this technique can be found in [Nayef and Mook 79].

Consider a solution to equation A.1 of the form:

$$x = A(t)cos(t + \phi(t)) \tag{A.2}$$

$$\dot{x} = -A(t)sin(t + \phi(t)). \tag{A.3}$$

The time derivative of the first equation must equal the second equation. This is the condition of compatibility and it implies:

$$\dot{A}cos(t + \phi) = A\dot{\phi}sin(t + \phi). \tag{A.4}$$

Taking the second derivative of x and substituting the result for $\ddot{x}$ into the equation of motion gives:

$$-\dot{A}sin(t+\phi) - A\dot{\phi}cos(t+\phi) + \hat{\mu}\, sgn\,(Acos(t+\phi)) - \mu\, sgn\,(Asin(t+\phi)) = 0. \quad \text{(A.5)}$$

Let $\gamma = t + \phi$, then the equation of motion and the compatibility equation can be written in the form:

$$\dot{A} = \hat{\mu}\, sgn\,(Acos(\gamma))sin(\gamma) - \mu\, sgn\,(Asin(\gamma))sin(\gamma) \quad \text{(A.6)}$$

$$A\dot{\phi} = \hat{\mu}\, sgn\,(Acos(\gamma))cos(\gamma) - \mu\, sgn\,(Asin(\gamma))cos(\gamma). \quad \text{(A.7)}$$

Averaging $\dot{A}$ over a cycle, assuming that both A and ϕ change slowly with respect to the frequency of oscillation, gives:

$$1/2\pi \int_0^{2\pi} \dot{A}d\gamma = 1/2\pi \int_0^{2\pi} (\hat{\mu}\, sgn\,(Acos(\gamma)) - \mu\, sgn\,(Asin(\gamma)))sin(\gamma)d\gamma \quad \text{(A.8)}$$

$$\dot{A} = -sgn\,(A)2\mu/\pi. \quad \text{(A.9)}$$

This result shows that the system is stable even for $\hat{\mu}$ greater than μ. This result also shows that only the friction in the system reduces the amplitude of the oscillation, i.e. the friction in the system is responsible for the decrease in energy of the oscillation. The intuition for this result can be found by observing that for a complete cycle the compensation $\hat{\mu}\, sgn\,(x)$ takes out the same amount of energy that it puts in, while friction always removes energy.

Averaging $\dot{\phi}$ over a cycle, assuming that both A and ϕ change slowly with respect to the frequency of oscillation, gives a similar result:

$$\dot{\phi} = 2\hat{\mu}/\pi|A|. \quad \text{(A.10)}$$

This shows that the compensation has the effect of changing the phase of the oscillation.