# AUTONOMOUS TACTILE OBJECT EXPLORATION AND ESTIMATION USING SIMPLE SENSORS

by

James G. Hollinger

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

J. S. Bay, Chairman

H. F. VanLandingham

A. L. Abbott

May, 1994

Blacksburg, Virginia

AUTONOMOUS TACTILE OBJECT EXPLORATION AND ESTIMATION USING SIMPLE SENSORS

by
James G. Hollinger

Committee Chairman: J. S. Bay
Electrical Engineering

(ABSTRACT)

In order for robots to become more useful, they must be able to adapt and operate in foreign or unpredictable environments. The goal of this thesis is to present an algorithm that will enable a robot to autonomously explore its environment by touch and then estimate the shape of objects it encounters. To demonstrate the feasibility and functionality of such an algorithm, it was fully implemented on a MERLIN 6540 industrial robot. A unique compliant end-effector (consisting of a trackball mounted to a force/torque sensor on a sliding mechanism) and a fuzzy logic force controller were developed to overcome the difficulties inherent in force control on a stepper motor robot. A Kalman filter based quadric shape estimator was then used to describe the objects encountered in the MERLIN's workspace. The minimization of a cost function based on the shape estimator's uncertainty guided the robot along an exploration trajectory designed to produce the fastest converging shape estimate. Results of various exploration trials using autonomous and preprogrammed trajectories are presented. In addition to shape estimates, surface curvature measurements were also obtained. The unique end-effector that provided compliance for the force controller was also able to measure the arc length traversed on the object's surface. Arc length combined with surface orientation makes it possible to determine local surface curvature.

## ACKNOWLEDGEMENTS

I wish to express my appreciation to the faculty of the Virginia Tech Bradley Department of Electrical Engineering for doing a fine job and special thanks to my advisor John S. Bay for his patience and guidance during the research and writing periods of this thesis. I am also indebted to Dr. Von A. Jennings of Martin Marietta for his generous contribution of a MERLIN 6540 Industrial Robot. None of this work would have been possible without it. A portion of this research was funded by the Office of Naval Research under grant no. N00014-91-J-1621.

## TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

## FIGURES

## TABLES

## 1.0 INTRODUCTION

Robot technology has advanced to the point where robots can operate autonomously in structured environments. An example of a structured environment would be an automated assembly line. In this type of environment, all objects are located in predetermined locations and all dynamic operations have been preprogrammed. Assembly line robots have a limited degree of adaptability: they can make minor adjustments to grasp a part if it is not in its exact predetermined location or they can identify a tool from a group of known tools. In general though, commercial robots do not have the ability to adapt to significant and unmodeled changes in their environment.

One of the present goals of the robotics industry is to enable robots to operate in unstructured/unmodeled environments. Such applications include hazardous waste removal, inspection and repair of nuclear power plants, fighter aircraft refueling, space station assembly, and even planetary exploration. Teleoperation has been investigated to accomplish some of the land based and low earth orbit applications. However, the need for a remote operator defeats the purpose of automation. An alternate approach would be to endow robots with the ability to sense and autonomously interact with their environment. Active areas of research in autonomous robotics include navigation, path planning, object recognition, grasp planning, and artificial intelligence. This thesis is primarily concerned with shape exploration and estimation - a subset of object recognition.

Object recognition is the process of examining an object, identifying useful pieces of information about its surface and then matching this information to known object models or using it to produce a condensed geometric description of the object. Relevant information includes, but is not limited to the three dimensional location, normal direction, and curvature of sections of the object's surface.

Information can be gathered with a number of devices such as stereo vision systems, magnetic resonance imagers (MRIs), laser and ultrasonic range-finders, and tactile probes. The resulting data is either densely or sparsely distributed over the object's surface and is acquired in unison or sequentially.

Stereo vision systems, MRIs, and range-finders that systematically scan a surface all produce a detailed 3D image of an object. The mass of digitized data that makes up the image can represent any one of a number surface attributes such as color, reflectance, shading, texture, and depth. Large amounts of meaningless and extraneous data are also found in the digitized image. The one thing that is common to most all images is that they contain information that is uniformly distributed over the object's surface. In order to realistically represent an object, the sampled points must be tightly (*densely*) packed together so critical details such as edges or holes will not be missed. This inevitably results in massive amounts of data. Unfortunately these methods of gathering data can also be sensitive to calibration errors, surface properties such as diffusion and reflection [33], lighting, and viewing angles. Another complication is that in most instances only a surface's location can be sensed. Normal direction and curvature must be calculated.

Tactile probes on the other hand gather data sequentially at a *sparse* set of points on the object. It is usually possible to guide or direct these probes as they gather data. Thus it would obviously be beneficial to develop an intelligent algorithm to aide the probe in its search for data. An algorithm could use previous data points to determine where the most interesting or beneficial data could be obtained next and try to avoid uninteresting or dull spots. We will call the resulting trek over the object's surface an exploration trajectory. A well executed exploration trajectory might be able to determine surface characteristics comparable to those determined by a dense data gathering system but with far fewer samples. In addition to surface location, most tactile probes are capable of directly measuring the surface's normal direction. Some artificial fingers even have the capability of measuring surface curvature [9] [15] [32] [40]. This means that each surface sample can contain multiple pieces of information as opposed to the single surface attribute captured in an image. Thus tactile probes gather data much more efficiently.

In this research we use an industrial trackball mounted to a force/torque sensor as a tactile probe. The trackball provides a frictionless single-point contact and enables the force/torque sensor to measure the surface normal. An industrial robot is used to position and orient the probe. In order to efficiently gather tactile data and explore an object's surface, an optimal exploration trajectory will be generated on-line. The trajectory will be optimized by using the tactile data to estimate the geometric shape of the object. As the estimate evolves, the probe can be guided to explore regions that will most effectively improve the estimate.

## 1.1 SURVEY OF LITERATURE

Object recognition methods are usually based on one of two techniques: surface modeling or solid modeling. Surface modeling techniques are best suited for sensors that gather dense data covering the entire object. After these sensors capture an image of the object, it is used to construct a number of low order surface patches. Second order or quadratic surfaces are usually used as patches because they possess the minimum order necessary to encode surface curvature. After enough patches have been constructed, they can be pieced together to form a complex surface resembling the original image. Segmentation can then be used to place boundaries around surface regions that have different attributes. Different types of surface patches can be used to cover the different segmented regions or the number of patches can be varied between regions based on their attributes.

One popular criterion for segmentation is based on surface curvature. A region with either constant positive or negative curvature can theoretically be described with a single quadratic surface patch. A zero crossing of surface curvature would signal a segmentation boundary and a new surface patch would be used on the other side. The segmentation method of surface modeling has several advantages over traditional methods that cover a surface with uniform patches. First of all the segmentation method is more efficient: fewer patches are required to produce the same quality surface model. Secondly, it reproduces the surface detail more accurately instead of smoothing it over. Recognizing the actual shape of an object is difficult if not impossible when using surface modeling because there is no underlying framework, only a skin exists. Sometimes it is possible to use a surface model to uniquely match the target object to a predetermined object in a database. However, this technique says nothing about new objects not already stored in the database.

Before surface modeling examples are cited in current literature, the second modeling technique (solid modeling) will be briefly described so that the two can be compared and contrasted in the cited literature. Tactile probes and other sensors that gather sparse data about an object's surface lend themselves to solid modeling. This is because they do not cover every inch of an object's surface. The geometric structure that is the framework of the solid model fills in the unexplored regions. Typical solid models include spheres, ellipsoids (quadrics), deformed ellipsoids (super quadrics), cylinders, cones, cubes, and polyhedra.

### 1.1.1 IMAGE BASED METHODS

Imaged based methods are quite different from the tactile exploration method of shape estimation described in this thesis, but they do provide insight into the usefulness of surface curvature and contrast the difference between object recognition based on matching and recognition based on shape estimation. A brief summary of image based methods follows.

### 1.1.1.1 SURFACE MODELING

The first set of image based methods cited uses surface modeling to produce a mathematical description of the image. Four of the five methods use surface curvature properties to assist in segmentation. Only one method attempts object recognition.

Sander and Zucker [38] perform surface modeling on a 3D image (obtained through magnetic resonance imaging) with uniformly sized quadratic surface patches. A least squares method is used to fit the patches to the 3D data. After the patches are pieced together, their second order property is exploited by identifying parabolic lines (lines of zero curvature) and partition the object into sections of positive and negative surface curvature. The authors indicate that quadrics or superquadrics could be fitted to the sections, but unfortunately make no attempt to do so.

Fan *et al.* [13] [14] locate jump boundaries, folds, and ridge lines in a 3D image using the zero-crossings and extrema of curvature. Jump boundaries are defined as points where the surface undergoes a discontinuity. Folds are characterized by discontinuities in the surface orientation and ridge lines correspond to smooth extrema of curvature. These features are used to segment the image. Surface modeling is then performed by fitting quadratic surface patches to the segmented regions.

Sinha and Besl [41] propose a similar technique for surface modeling. Lines of principle curvature to segment a 3D image before fitting quadratic surface patches to it.

Arman and Lee [3] also use surface curvature to segment a 3D image. Lines of Gaussian curvature and mean curvature are used to locate linear surface patches on an object. The edge length, angles and surface area of the patches are then used to match the object to a database of polyhedra. The polyhedra are generated beforehand by a commercial Computer Aided Design (CAD) system. No provisions are made for describing any new objects encountered.

Delingette *et al.* [11] perform a cross between surface and solid modeling. They model 3D images by globally deforming a tessellated icosahedron (a solid object constructed from triangular faces.) Data points obtained with a range-finder are used to deform the more than 500 triangular faces. Although this method uses a geometric model, the large number of faces and associated surface parameters inhibit shape recognition.

#### 1.1.1.2 SOLID MODELING

The next set of image based methods use a single equation to describe the imaged object. This type of solid modeling can provide useful shape information about an object such as volume and axis length. The parameters that make up a solid model can be used to match the imaged object to a database of previously modeled objects.

Terzopoulos and Metaxas [45] use a unique type of deformable superquadric to model an object in a 3D image. Their superquadric is dynamically deformed through iterations using simulated forces. Because the model is locally deformed, 1000 to 2000 nodes must be distributed over the model's surface to obtain enough flexibility. The simulated forces, derived from the distance between the image points and the model's surface, are applied at the nodes. Typically 200 iterations are required before the model's surface will reach equilibrium. This method is really a cross between solid modeling and surface modeling because it attempts to reproduce the fine detail of the object by distributing nodes over its surface. Because of this characteristic it suffers from an excessively large number of parameters. The excessive number of parameters result in a non-unique solution which makes object recognition difficult.

Sclaroff and Pentland [39] propose a similar technique that uses a finite element method to dynamically deform a solid model. The advantage of the finite element method is that the degrees of freedom of the model can be chosen arbitrarily. This is possible because the method deforms the model globally instead of locally. This enables the object modeling problem to be formulated in an over constrained fashion and facilitates object matching. However, the distributed deformations still make it difficult to describe the resulting shape.

Pentland [34] [35] performs solid modeling by extracting multiple deformed superquadrics from 3D images. The 14 parameters describing each of the superquadrics are determined from a linear regression of the surface points. The recovered object model is then checked by using computer graphic techniques to render an image of the model and comparing its histogram to that of the real image. No discussion is presented to describe the number, location, and orientation of the superquadrics needed to accurately model the object.

Solina and Bajcsy [42] match a single deformable superquadric to an object in a 3D image by minimizing a cost function. The value of the cost function depends on the distance of the image points from the model's surface and the overall size of the model. Fifteen parameters are used to describe the globally deformable superquadric. Typically 30 iterations are required to obtain a good fit. The paper points out that deformed superquadric models are ambiguous (different parameter sets can describe the same shape,) but a unique solution can be obtained by constraining the search to a portion of the parameter space.

### 1.1.2 EXPLORATION METHODS

The remaining literature is concerned with object recognition techniques that employ sensors to collect surface data sequentially. The first group of techniques perform blind exploration. They make no attempt to use previous data to influence future probings. The second group of techniques explores objects intelligently.

#### 1.1.2.1 UNGUIDED EXPLORATION

Kofakis [24] proposes to use a set of points, consisting of location and normal direction, scattered over an object's surface to identify it from a set of known objects. In addition, the object's location and orientation will also be determined. The location of a point on the object's surface is obtained with a 3D laser range-finder. The normal direction at that point is found by measuring the location of two nearby points and computing the cross-product of the directional vectors from the original point to the two new points. A nonlinear least squares minimizing algorithm is executed after a new point is found. This algorithm finds the best location and orientation of each known object that matches the gathered data. Eventually enough points will be collected to eliminate all but one of the known objects. No attempt is made to evaluate where beneficial data might be found next. The exploration is assumed to be a regular (or random) scan of the object as it is rotated in the robot's grasp.

Gaston and Lozano-Pérez [17] and Grimson and Lozano-Pérez [18] also discuss how to identify and locate an object from a set of known objects. Using only the contact location and surface normal, a tree of consistent hypotheses is successively pruned. A hypothesis is consistent as long as the sensed points match a known object in a possible orientation and location. Again, known objects are eliminated by probing the surface at new locations, but no attempt is made to optimize the process.

Allen [1] describes three exploratory procedures and their associated object models for determining attributes of an object. These procedures were based on studies of human tactual perception. All three were implemented on a Utah-MIT hand mounted on a PUMA 560 robotic arm. The first procedure (also described in [2]) is termed "grasping by containment." It is useful for extracting superquadric models from an object. Contact points on an object are generated by repeatedly closing the robotic hand around an object. Successive measurements are obtained by moving the hand along a predetermined trajectory. A Levenberg-Marquardt nonlinear least squares estimator is used to fit a superquadric to the data. Eleven parameters are used to describe the superquadric (13 if the object is tapered at its ends.) Typically 30 to 100 contact points were acquired.

The second procedure is termed "lateral extent." It is used to determine the face, edge, and vertex of a portion of an object. This is accomplished by repeatedly touching the surface with a single finger until an edge is found and then following the edge until its vertex is found.

The third procedure is called the "contour follower." It is useful for extracting generalized cylinder models from an object. A vision system is used to locate the object's axis before exploration is begun. The radius of the generalized cylinder is determined by pinching the object between the thumb and index finger of the robotic hand. The object is explored by releasing it and moving the hand along its predetermined axis before pinching it again. None of the three procedures make an attempt at using previous sensor data to guide the hand. Thus, a blind exploration is performed.

### 1.1.2.2 INTELLIGENT EXPLORATION

The following techniques attempt to optimize the data they gather so that their goal can be obtained in as few steps as possible. This is accomplished be considering old data and evaluating the benefit of new potential data. The first technique in this group does not restrict itself to a continuous exploration trajectory. Instead it haphazardly jumps around the object. The second technique performs a number of discontinuous probings that create a zig-zagging trajectory. The final technique performs a qualitative exploration based on human perception.

Cameron [10] uses statistical decision theory to position a tactile sensor in the robot's workspace. This method minimizes the number of sensed points required to identify an object from a set of known objects. Probabilistic membership functions are constructed in the workspace based on the known object shapes and the sensed data. Utility functions are then used to qualitatively describe the value of obtaining new data at every location in the workspace. The value of obtaining data at a specific location depends on how useful it would be at constraining the object's shape. These two functions are combined to make a Baye's decision as to where the best location in the workspace would be to probe. This method has the distinct disadvantage of making a tactile probe repeatedly make and break contact with an object. Making contact with an unmodeled object is a delicate and time consuming process. Data can be collected more quickly and safely by maintaining contact and sliding along the object's surface to reach new regions. The autonomous exploration algorithm described in this thesis uses a fuzzy logic force controller to minimize the number of times contact is lost and must be reacquired.

Lee and Hahn [29] use an optical proximity sensor (capable of measuring depth and orientation) to uniquely discern an object from a set of known object models. Each object model is constructed solely of quadrics. These include planes, cylinders, spheres, and conic sections. The goal of this method is to identify and locate an object with a minimum number of surface probings. This is accomplished in two steps. First a set of hypotheses is constructed that represents all the possible orientations the known object could be in that would still match the collected data. The authors refer to these hypotheses as a multiple interpretation image. In the second step an optimal probing direction is selected. This direction will provide the minimum ambiguity among the possible interpretations when it is followed. While this technique can identify quadric objects, it does not determine the parameters that give them their shape. This must be done beforehand for each of the objects that might be encountered.

Roberts [37] uses continuous touch to identify objects from a set of known polyhedra. The unknown object is explored with search sequences such as:

> "Beginning on or near a face, trace across the face in a chosen direction until reaching an edge. Move just far enough on the edge to acquire its direction. Move just far enough out onto the face beyond the edge to acquire its normal, and then end up near that edge."

Before a move is made, all possible search sequences are assigned an efficiency value. This value is based on the number of possible models that can be removed if the combination of moves can be completed successfully. The sequence with the greatest efficiency is then chosen and executed. Unfortunately this algorithm is designed to eliminate inconsistent models from a set of known models. It does not try to determine the shape of the object. In addition, it provides no information about a new object other than the fact that it is not in the database.

### 1.1.3 THEORETICAL BASIS

The theoretical basis for this thesis comes from Bay and Hemami [7]. In this paper an evolving model of an unknown object is used to improve force tracking on the object's surface while following a predetermined trajectory. An extended Kalman filter is used to iteratively update an estimate of the object model. The inputs to the Kalman filter are the contact location between a tactile probe and an object's surface plus the local surface normal. Force tracking results are presented for a simulated ellipsoid.

Further simulated results are presented in [8]. A simulated multifinger robotic hand is used this time. The trajectory of the fingers was constrained to the surface of a simulated ellipsoid while the hand was rotated as a unit around a fixed axis. The convergence of the ellipsoidal estimate is compared for one- through five-fingered hands. The conclusion drawn is that normal measurements greatly improve the estimate. Additional fingers are also beneficial, but not nearly as much as normal measurements. The paper also describes a linear formulation of the surface normal with respect to the nine object parameters. This formulation, combined with a simple linear formulation of the contact location allows a linear Kalman filter to iteratively update the object model estimate.

In [5] and [6] a cost function is derived based on the Kalman filter estimate's covariance matrix from the previous method. It is shown that this cost function is a measure of the uncertainty of the object model's estimate and it is also an implicit function of the surface measurements (contact location and surface normal.) This last property is used to calculate a search direction that will minimize the cost function: a direction that, when followed, should cause the object model's estimate to converge faster than any other. Again, noiseless, simulated exploration results are presented for a quadric surface. A simulated exploration of a superquadric is also included in [5].

The experimental exploration and estimation results contained in this thesis are presented in [20] as well as the results of a technique to measure a surface's radius of curvature. These results were obtained by implementing the theory of the previous method on an industrial robot. This was done to investigate a number of aspects related to real world work such as sensor design and construction, and controller requirements, tuning and performance. In addition, a number of questions related to shape estimation were looked at such as how robust was the shape estimation algorithm, what were its rates of convergence, and what degree of sensor accuracy was needed.

## 1.2 STRUCTURE

The goal of this thesis is to investigate the problems, requirements, properties, and performance related to the real world implementation of an optimizing autonomous exploration and shape estimation algorithm on an industrial robot. It will attempt to address the deficiencies of past literature, namely the lack of an intelligent exploration algorithm resulting in a shape description (most literature matches the object to a pre-existing database). A quadric solid model will be used to provide a flexible description of the object's shape. The exploration process will be optimized to produce the fastest converging shape estimate possible.

An overview of autonomous, tactile shape exploration and estimation is presented in the first chapter. Chapter One also includes a survey of related literature. The industrial robot, controller, electrical interface, compliant end-effector, and sensors used to perform the research are described in Chapter Two. This includes their specifications and integration into the tactile exploration system. Chapter Three discusses the requirements, considerations, and control algorithms used to perform tactile exploration. A fuzzy logic force controller is described in this chapter just after a cursory discussion of fuzzy logic control is presented. In addition, the mechanics and dynamics of tactile exploration are examined. Shape estimation is covered in Chapter Four. This includes the model used to describe the object's shape, the statistical filter used to estimate the parameters that determine the model, and the autonomous exploration routine. Surface curvature and the method used to measure it are also discussed in this chapter. The experimental results obtained while researching this thesis are presented in their respective sections. The meaning and implications of this thesis are discussed in Chapter Five, the conclusion. Directions and motivation for further research are also discussed.

## 2.0 HARDWARE

This chapter describes the experimental setup used to physically implement the theoretical portion of this thesis. The setup consists of a MERLIN 6540 Industrial Robot, its controller, an Intel 80386 Personal Computer (PC), a customized controller/PC interface, and an assortment of unique sensors.

The MERLIN robot is a six degree of freedom robotic arm. It is capable of arbitrarily positioning and orientating an end-effector in its workspace. It has a 40 inch reach and a 50 pound payload capacity. The arm is constructed with six stepper motors. Microstep motor controllers give 25,000 stable shaft positions per motor revolution. A 24:1 gear reduction increases the accuracy and torque of the arm. The combination of optical encoder and gear reduction give an axis resolution of 0.0075 degrees per revolution and a Cartesian accuracy of approximately 0.005 inches. This robot was designed to operate on an assembly line and has the capability of being mounted up-side-down to an overhead gantry. The work described in this thesis requires an accurate positioning device with a soft touch. Therefore a customized controller and a unique end-effector had to be developed for the MERLIN.

## 2.1 CONTROLLER

The standard MERLIN controller consists of nine microprocessors operating concurrently in a Versabus backplane. They can be functionally grouped as follows:

- Main processor
- Input/Output processors
- Axis Control processors

A block diagram of the system is presented in Figure 2.1. The main processor is a 32 bit Motorola 68000 operating at 8MHz. This processor is tasked with coordinating the motion of the six axes. It can be programmed through two interpretive languages: BASIC and AR-Smart (an unsophisticated language programmed via a hand-held pendant). Alternatively it can execute a High Speed Host Interface (HSHI) program.

There are two 8 bit Motorola 6809 microprocessors that supervise all the controllers input and output (I/O). They monitor the front panel pushbuttons, pendant switches, and digital inputs. In addition they assert the digital outputs, control analog to digital (A/D) conversions to monitor the pendant and analog inputs, control digital to analog (D/A) converters, and control the disk drives. Data is transferred to and from the main processor via shared random access memory (RAM) on the two microprocessor I/O boards.

The six remaining microprocessors are also 8 bit 6809s. They are equally divided between two boards. Each microprocessor controls an individual axis of the robot. The processors operate in 4ms cycles. At the beginning of a cycle a processor reads the desired motor position (in encoder counts) from on-board shared RAM. It then compares this to the current position of the motor (maintained by encoder circuitry on the board). A microstep position profile is then computed and executed that should rotate the motor

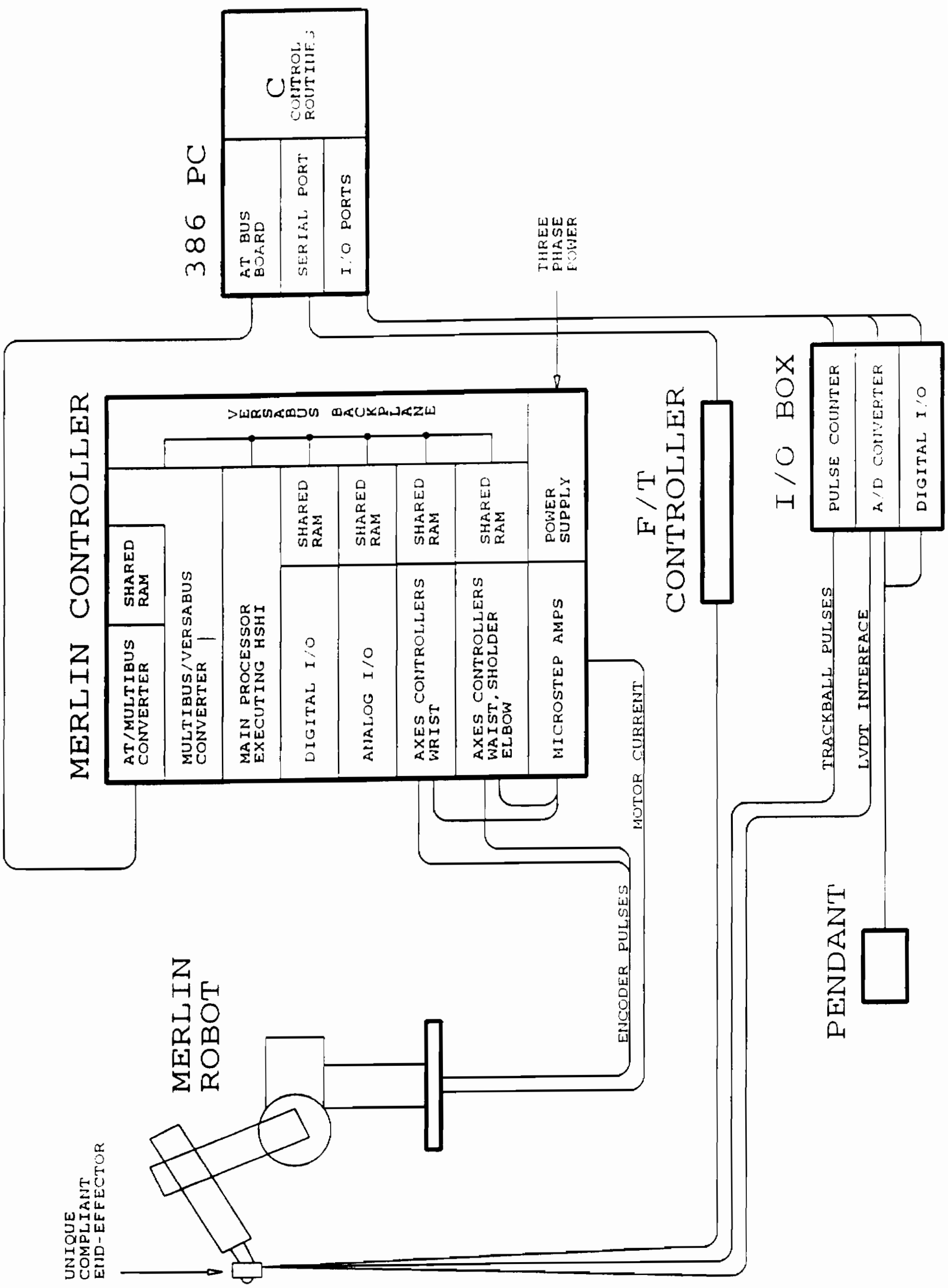


**Figure 2.1** Block diagram of robot/PC interface.

to the desired position by the end of the 4ms cycle. At the end of the cycle the new motor position is transferred to the shared RAM. When the main processor is executing a BASIC or AR-Smart program it deposits a set of six motor positions into the axis controller board's shared RAM after it has computed the desired end-effector position. If the main processor is executing HSHI it will copy the six desired motor positions from user supplied shared RAM to the axis controller board's shared RAM every 4ms. This interface provides a means by which the robot can be controlled from an external computer.

The HSHI interface was implemented by installing a Versabus/Multibus converter board in the MERLIN controller's backplane. An IBM AT/Multibus converter was then plugged into the converter board. The AT/Multibus converter consists of a Multibus board and an AT bus board connected by two shielded ribbon cables. The Multibus board contains 32K of shared RAM and was installed in the Versabus/Multibus converter located in the controller's backplane. The AT bus board was installed in a 33MHz Intel 80386 PC. Customized routines written in C on the PC placed desired motor positions in the shared RAM on the Multibus board. The HSHI program then copied the motor positions from the Multibus shared RAM to the axis controller's shared RAM. This allowed the robot to be controlled from the PC.

## 2.2 SENSORS

There are three important sensors connected to the robotic arm's wrist (see Figure 2.2). The first is a wrist mounted force/torque sensor. The controller for the sensor can decompose an arbitrarily applied force into three orthogonal forces and torques about a user defined coordinate system. The controller senses the applied forces and torques via eight strain gages located inside the sensor. Decomposition and transformation computations are performed on the strain gage readings every 10ms. The resulting forces and torques are transmitted serially to the PC at 19200 bits per second.

An industrial trackball is also attached to the wrist. It is used to measure distances and surface curvature, the details of which will be discussed in Chapter Four. The trackball's housing is mounted to the force/torque sensor by a unique sliding mechanism. The mechanism introduces a variable amount of compliance into the end-effector. It consists of four springs and smooth posts that the trackball housing can slide on. The compliance can be adjusted by changing the stiffness of the springs. The ability to control the compliance of the end-effector is critical to implementing a stable force controller on a stepper motor robot. A sufficiently compliant end-effector will transform changes in position into changes in applied force when the robot is in contact with a hard surface. The details of force control and compliance will be discussed in Chapter Three.

Inside the trackball housing are two optical encoders that measure the rotation of the trackball. A pulse counting card connected to the PC detects the output of the encoders and maintains the trackball's position. The interface between the PC and the

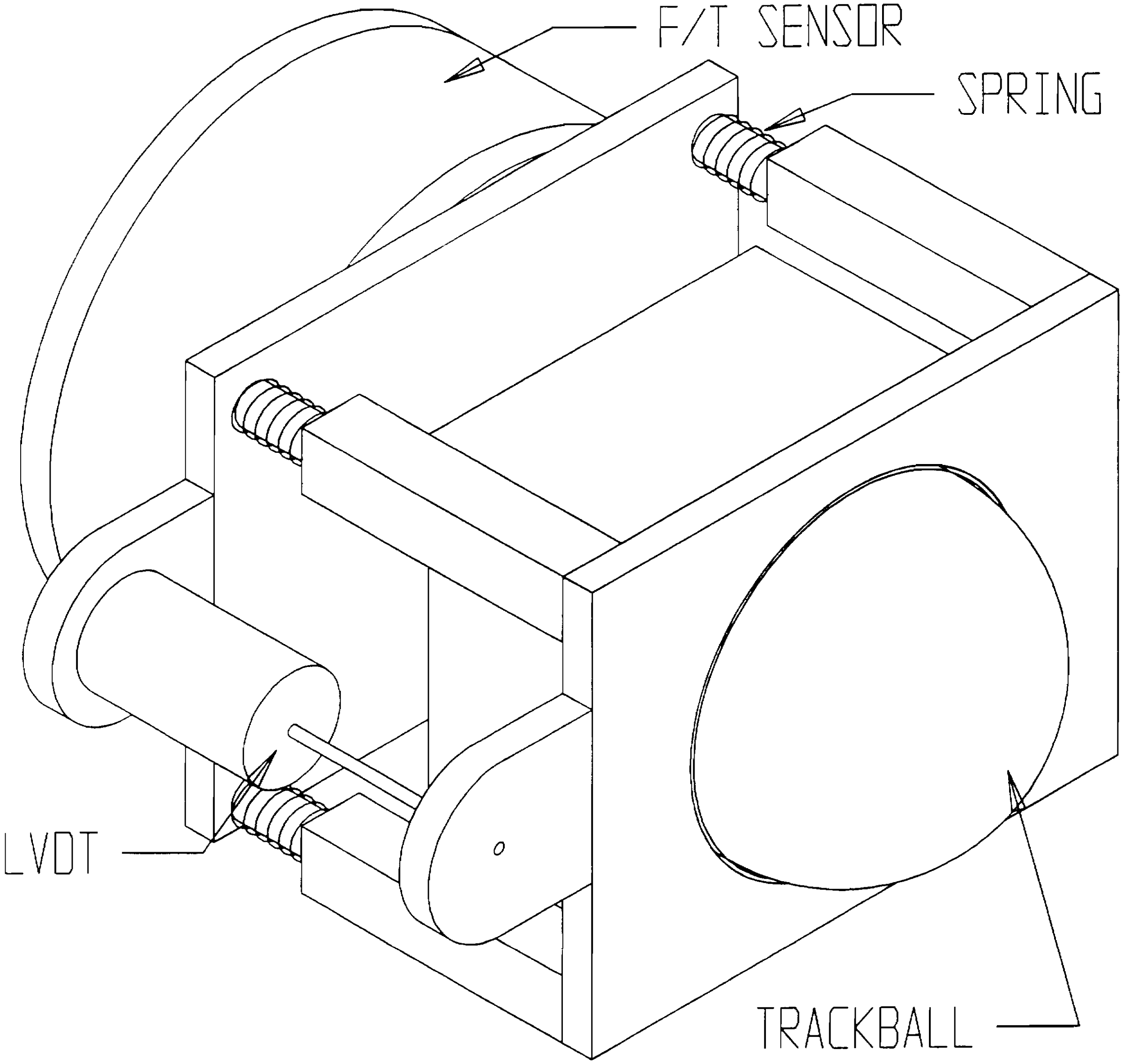


**Figure 2.2** End-effector attached to robot's wrist.

pulse counting card is through several I/O ports on the AT bus. The encoders inside the trackball housing generate 500 pulses per revolution of the trackball. The trackball is 2½ inches in diameter which results in a position accuracy of 0.0157 inches.

The compliance built into the end-effector makes it difficult to determine the trackball's true location when it is in contact with an object. This is because the relative location of the trackball with respect to the force/torque sensor changes as contact forces compress the four springs. A third sensor, a Linear Variable Displacement Transducer (LVDT), was added to the end-effector to measure this displacement. The LVDT was mounted between the force/torque sensor and the trackball housing. An A/D card connected to the PC interfaces with the LVDT in a manner identical to the pulse counting card. Figure 2.3 shows a plot of data used to calibrate the LVDT. The plot clearly shows the linear relationship between displacement and the output of the transducer. By monitoring the LVDT, the kinematic equations of the robot can be continuously updated to compensate for the compliance of the end-effector.

The combination of previously described sensors forms a unique tactile sensor. It not only provides compliance to enable the robot to control contact forces, but it measures the contact location, local surface normal and the arc length traversed while the robot explores an object's surface. The PC-MERLIN controller interface greatly increases the computational power and flexibility of the robot. A wide assortment of sensors can be interfaced with the PC and incorporated into complex control algorithms. The following two chapters describe how the robot and its controllers and sensors were used to autonomously explore its workspace and efficiently estimate the shape of the objects it encountered.

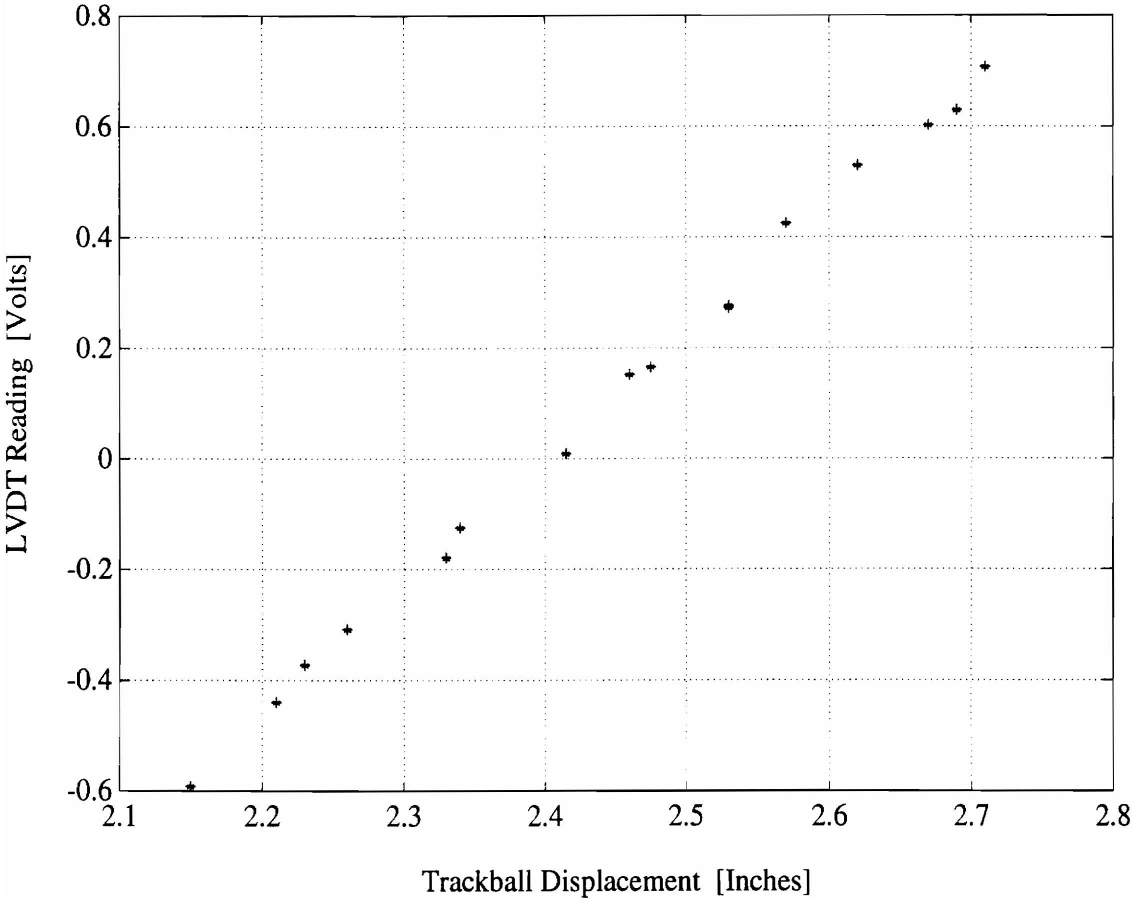


**Figure 2.3** LVDT calibration data.

## 3.0 SURFACE TRACKING

Tactile exploration relies on the ability to gather data about an object through physical contact. In order for a robot to accomplish this it must be able to make initial contact with the object. Unless the tactile sensor's contact area covers a large percentage of the object, not much information can be obtained through a single contact. Thus, the robot must reposition itself over the object to obtain new and interesting information. If the robot simply retracts from the object, moves to a new location, and regains contact, much information about the object will go undetected. Specifically, how the object's surface changes from one point to another (*i.e.* its curvature) will be missed. Therefore, a robot attempting to explore an object with tactile sensors must posses a means by which to make initial contact with the object and maintain contact as it moves its tactile sensors.

This chapter will describe how a MERLIN robot has been endowed with the capabilities discussed above. First, a method for making initial contact will be described. Next, a method for maintaining contact as the robot moves its end-effector around the object is presented.

A force controller is needed to regulate the applied force while the robot is in contact with the object. It will be shown that a fuzzy logic controller is well suited for this task. The final section of this chapter will describe the implementation of a fuzzy logic force controller on the MERLIN robot [21].

## 3.1 CONTACT

Without global knowledge of the environment or a vision system to locate objects and guide the end-effector to them, it will be assumed that the manipulator is initially in a configuration that places the end-effector reasonably close to the object. Additionally, the end-effector should be oriented such that it points towards the object. Contact can then be achieved by performing a *guarded move*. This involves advancing the end-effector toward the object until an increase in force is detected. The rate of advance should be constrained so that after contact is detected, the motion of the manipulator can be halted. Otherwise the manipulator would crash into the object. This would not only present the possibility for damage to the manipulator, but could momentarily deform the surface of the object and make it difficult to determine its true shape.

After initial contact has been made, the end-effector will be repeatedly repositioned on the surface in an attempt to gather additional information. The subject of where to reposition the end-effector will be discussed later. For now, it will be assumed that a search direction tangent to the surface has been determined. The end-effector must then be moved in this direction. Since it is also desirable to measure surface curvature, the trackball attached to the end-effector should be in contact with the surface and rotating during the entire move. These two requirements can be satisfied by dividing the move into a series of small steps. After each step, the contact force should be examined and adjustments made to correct it if necessary.

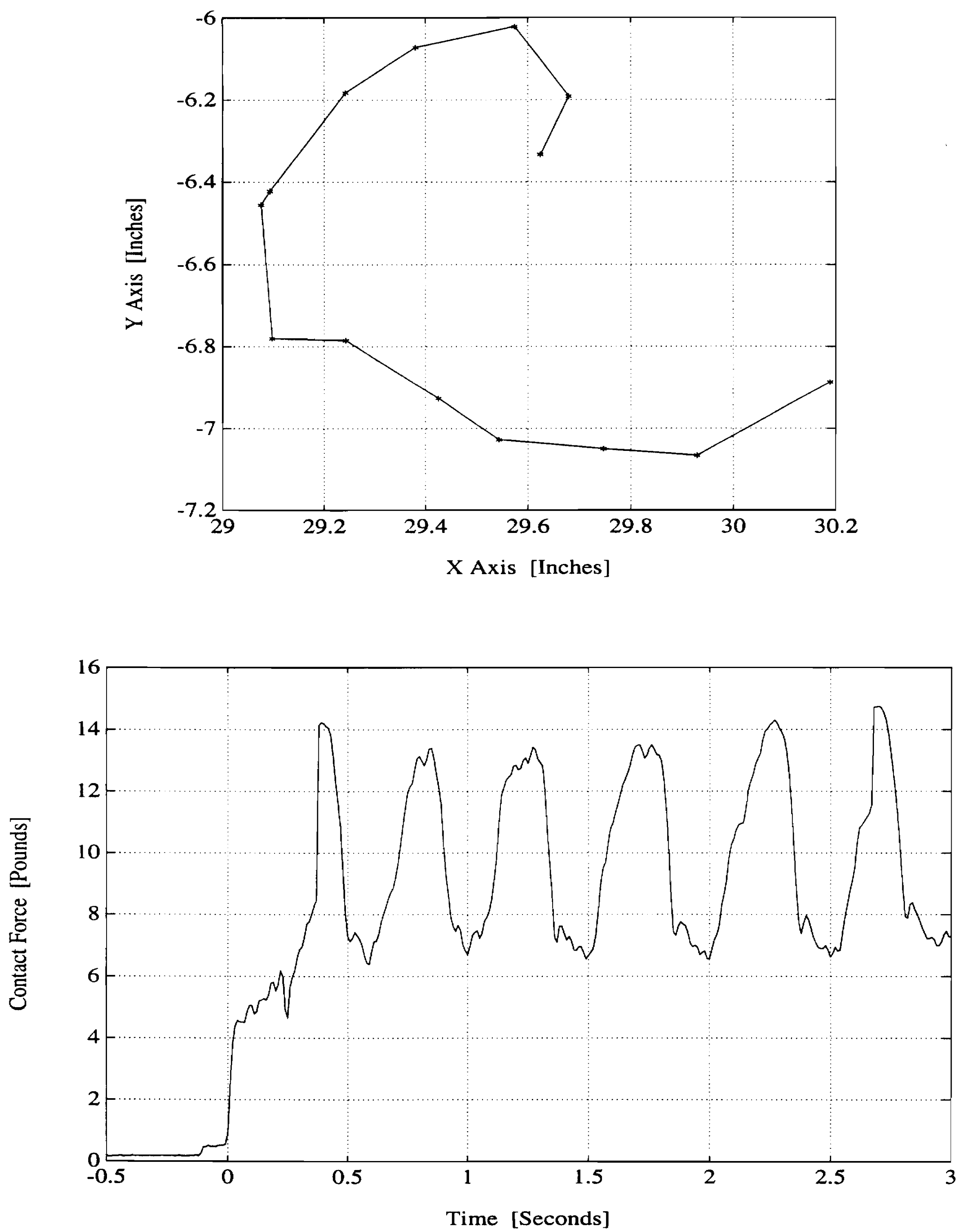


**Figure 3.1** Trace of end-effector on a table top and resulting contact force.

The experiments performed in this research used twenty steps, of 0.001 inches each, to complete a move, of 0.02 inches, while 8 to 12 pounds of contact force was maintained. Figure 3.1 shows a typical series of moves and the corresponding contact force. During the move, the desired position of the end-effector had to be stored and updated in software. This was necessary because the step size was less than the accuracy of the manipulator (0.005 inches). A step of 0.001 inches will not always cause the end-effector to move. Only after a series of 0.001 inch steps have been accumulated, will the end-effector make a detectable movement. Thus, if the desired position is reset to the actual position at the beginning of each step, the 0.001 inch increments will be lost and the end-effector will not move. Such a small step size was necessary to allow the force controller time to regulate the contact force. The total distance traveled in each search direction was 0.02 inches. This is well within the 0.005 inch accuracy of the manipulator.

The wrist mounted force/torque sensor measures the contact force along the three axes of the sensor. Since the trackball provides a near frictionless contact and eliminates tangential forces, the direction of contact force should be the same as the local surface normal. This provides the surface normal with respect to the sensor's coordinate frame. Knowledge of the kinematics of the manipulator and the current joint angles makes it possible to rotate the surface normal from the sensor's coordinate frame to the inertial coordinate frame. In addition to providing valuable information about the surface, the direction of the surface normal can be used to adjust the orientation of the end-effector so that it will remain perpendicular to the object's surface. This is advantageous because it keeps the trackball housing from colliding with curved surfaces.

## 3.2 FORCE CONTROL

A force controller is needed to prevent the contact force from becoming too large or too small. A large contact force could deform the object's surface, degrade surface measurements, cause the trackball to slip, and physically damage the sensors. Contact with the surface could momentarily be lost if the force became too small. This would degrade curvature measurements because the trackball can only rotate when it is contact with the surface.

Robotic force controllers can be divided into two categories depending on their mode of implementation. They can be classified as either computed torque or position-based force controllers. A computed torque controller derives the necessary information to control the joint torques from the desired end-effector forces and the dynamic model of the manipulator. In symbolic notation this can be expressed as [16, pg. 92]:

$$\tau = I(\theta)\ddot{\theta} + H(\theta,\dot{\theta}) + G(\theta) + J^{T}(\theta)\Gamma$$

where $H(\theta,\dot{\theta})$ and $G(\theta)$ represent centripetal, Coriolis and gravitational effects, $I(\theta)$ represents the manipulator's inertia matrix and $J^{T}(\theta)$ is the transpose of the manipulator's Jacobean. $\theta$ is a vector of joint angles, $\Gamma$ is a vector of the desired end-effector forces and $\tau$ is the resulting vector of joint torques. This method is most effective when the dynamics of the manipulator can accurately be determined [12], [26]. In addition, this method requires that joint torques be controllable.

Position-based force controllers are most useful for robots without the capability for joint torque control [47]. However, this method requires the contact force to be sensed. An error signal proportional to the difference between the desired force and measured force is fed back to the force controller and used to adjust the position of the manipulator's joints. Through this method, the manipulator can be made to behave like a mass-spring-damper with adjustable stiffness and compliance. This can be accomplished by controlling the Cartesian acceleration of the end-effector based on the sensed contact force [46], [22]. A more general survey of force control techniques can be found in [19] [30] [36] [48].

Elosegui, *et al.* [12] demonstrated that position-based force control is very sensitive to the compliance of the manipulator-environment connection. Kazerooni, *et al.* [23] and Wen [47] show that force stability cannot be guaranteed for a hard contact: when both the manipulator and environment have very little compliance. Instability can result in this instance because a small change in the end-effector's position will result in a large force reaction between the manipulator and the environment.

The MERLIN robot is constructed with stepper motors which make joint torque control impossible. Thus, a position-based force controller must be implemented. To avoid the instability problems associated with a hard contact, instrumented compliance was built into the end-effector. The compliance was achieved by placing four springs around the posts of a sliding mechanism used to hold the trackball. The springs give the end-effector a stiffness of approximately 4 pounds per inch. This is much less than the stiffness of the objects being explored and will prevent their surfaces from being deformed. Experiments revealed that the trackball rotated best when 8 to 12 pounds of contact force was exerted.

Another force control complication is that the kinematics and joint offsets of the MERLIN are not known exactly. To compensate for this uncertainty, the force controller should be insensitive to kinematic constants and joint offsets.

This can be accomplished by implementing a nonlinear force feedback controller. The controller should rapidly respond to large force errors but should reduce its sensitivity when the contact force approaches its target. Since the goal of the force controller is simply to maintain contact and prevent the trackball from slipping, the target force can range from 8 to 12 pounds. Thus, the controller should have minimal response for force errors of ±2 pounds, but should quickly react to larger force errors (errors that could cause the manipulator to loose contact or damage the object).

Fuzzy logic controllers are an ideal way to implement nonlinear control with nonspecific requirements. The desire to maintain an 8 to 12 pound contact force range instead of a specific fixed force value fits the description of a fuzzy logic controller much better than that of a traditional proportional-integral-derivative (PID) controller [27] [28]. The nonspecific force requirements should also make a fuzzy logic controller more robust to varying surface compliances than a PID controller.

## 3.3 FUZZY LOGIC CONTROL

A fuzzy logic controller maps inputs to outputs according to linguistic rules. An example of a linguistic rule would be:

**If the input is small, then the output should be slow.**

This would result in a slow output as long as the input was small. The inputs and outputs can be expressed as either integers or floating point numbers and usually have physical interpretations such as voltage, range, velocity, etc. They are commonly referred to as crisp values. Internally, a fuzzy logic controller represents the inputs and outputs as linguistic variables such as small, medium, tall, slow, fast, etc.

The mechanics of the controller can be divided into the following three distinct sections.

1) Fuzzifier
2) Inference Engine
3) Defuzzifier.

The first step, performed by the fuzzifier, converts the crisp inputs into sets of linguistic variables. Next, the inference engine uses a rule base to operate on the input sets and produce an output set. The final step, performed by the defuzzifier, converts the output set of linguistic variables into crisp outputs.

Every linguistic variable has a membership function associated with it. This function is used to determine the degree to which a crisp value belongs to the linguistic variable. In the fuzzification step, a membership value is assigned to each linguistic variable used to describe the input. For example, suppose the input was a person's height and the two linguistic variables used to describe it were tall and short. The following two membership functions would have to be defined:

$$\mu_{TALL}(\mathbf{height}) \in [0,1]$$

$$\mu_{SHORT}(\mathbf{height}) \in [0,1]$$

where $\mu_X$**(height)** is the membership function for linguistic variable **X** that maps the input, **height**, into a membership value from zero to one. Possible membership functions are shown in Figure 3.2 along with some additional membership functions that will be used in proceeding examples. Now, if a particular person was 5'9.6", they would be tall to degree 0.8 and short to degree 0.2.

The major distinction between Boolean logic and fuzzy logic has just been illustrated. In Boolean logic, a person can only be classified as either tall or short. However, fuzzy logic allows a person, who is not obviously tall or short, to be more realistically classified as tall and short to varying degrees.

The linguistic rules in the rule base have the form:

**If antecedent, then consequent.**

The antecedent describes a sufficient condition for the consequence to be true. The condition is only sufficient and not necessary because there may be other rules that also imply the same consequence. The antecedent of the example rule:

**If the input is small, then the output should be slow**

says that a small input will cause the consequence (a slow output) to be true. The degree to which the consequence is true equals the degree to which the antecedent holds. If the antecedent is simply an input belonging to a specific linguistic variable, then the truth of the consequence is the same as the membership value of the input in the specific linguistic variable:

$$\textbf{Truth of consequence} = \mu_X(\alpha).$$

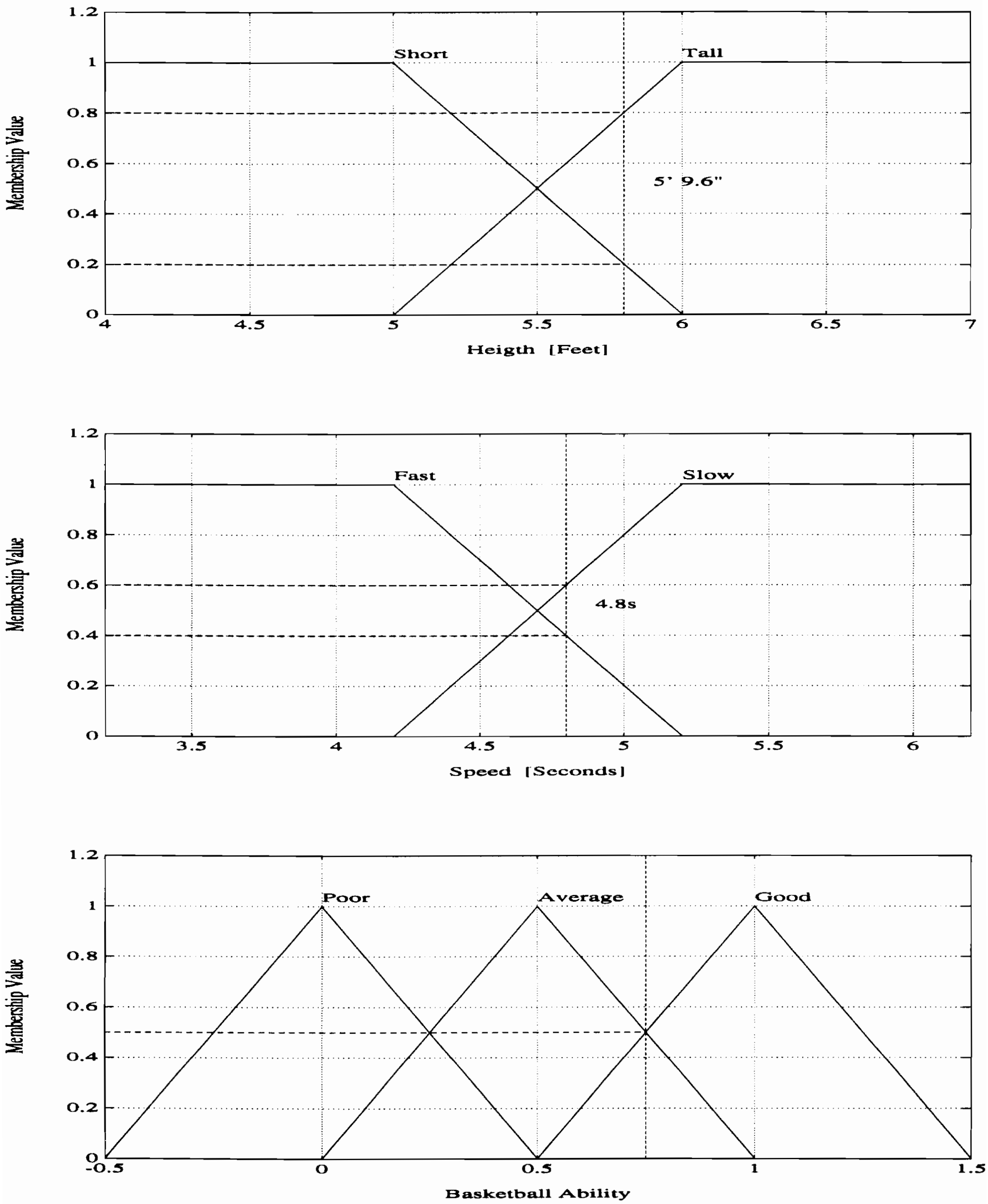


**Figure 3.2** Membership functions.

Several antecedents may be combined with "AND" operators in which case the degree to which the consequence is true equals the minimum value to which any of the antecedents hold. Similarly, "OR" operators may be used along with the maximum value to which any antecedent holds [25, Chap. 8].

The rules for a fuzzy logic controller are usually arranged in matrix form with a different matrix for each output. Every element of a matrix corresponds to a unique linguistic classification of the inputs and supplies the resulting rule consequence. For example, a two input, one output controller that uses two linguistic variables to describe each input would require a two-by-two rule matrix. Suppose the two inputs were a person's height and speed described by, respectively, tall/short and fast/slow. The output could then be used to describe their ability to play basketball (good/average/poor) via the following rule matrix.

**Table 3.1** Rule matrix describing basketball ability.

| Height | Speed: Fast | Speed: Slow |
|---|---|---|
| Tall | Good | Average |
| Short | Average | Poor |

The top, left element of the matrix would represent the rule:

**If the person is tall and they are fast,**
**then they should be a good basketball player.**

After evaluating all the rules in the matrix, the output will be described by a set of truth values for the linguistic output variables. The process of converting these truth values into crisp outputs is called defuzzification. The truth values can be used to scale the membership functions of the linguistic output variables, or they can be used to saturate the membership functions at maximum value. The former is called correlation-product encoding and the latter is correlation-minimum encoding. Correlation-product encoding preserves more information about the membership functions and will be used in the design of this fuzzy logic controller.

After scaling the membership functions by the appropriate amount, they can be graphically superimposed on each other. The centroid of the resulting conglomerate shape will be the crisp output value. If correlation-product encoding is used and the membership functions of the output variables are symmetric and unimodal, calculation of the centroid will reduce to the following summation [25, pg. 386]:

$$v = \frac{\sum_{i=1}^{N} w_i c_i A_i}{\sum_{i=1}^{N} w_i A_i}$$

where $w_i$ is the truth value of the $i_{th}$ rule's consequent; $c_i$ and $A_i$ are the respective centroid and area of the membership function associated with the $i_{th}$ rule's consequent. $N$ is the number of rules in the rule base and $v$ is the resulting centroid.

As an example of defuzzification, consider the two-by-two rule matrix previously presented but without the two rules resulting in average consequences. The two inputs and single output will still be, respectively, a person's height, speed, and their ability to play basketball (good/poor). A 5'9.6" person who can run the 40m dash in 4.8s would yield the following truth values for the two linguistic output variables.

**Table 3.2** Truth values.

| Linguistic Variable | Truth Value |
|---|---|
| Good | 0.6 |
| Bad | 0.2 |

The centroid of their scaled membership functions is:

$$\frac{0.6\times 1\times\frac{1}{2} + 0.2\times 0\times\frac{1}{2}}{0.6\times\frac{1}{2} + 0.2\times\frac{1}{2}} = \frac{0.6}{0.8} = 0.75$$

on a range from zero to one. Therefore, the defuzzified output is 0.75 and indicates that the person should be an above average basketball player. Figure 3.3 illustrates how these values were obtained. This is just an illustrative example and is not meant as a genuine test of basketball proficiency.

The three stages of a fuzzy logic controller can be summarized as follows: First, the crisp inputs are fuzzified by determining their degree of membership in the linguistic variables used to describe the inputs. The inference engine then uses a rule base to determine the degree to which linguistic variables used to describe the output are true. Lastly, the linguistic output variables are defuzzified by calculating the centroid of their scaled membership functions. The resulting, crisp value, is the output of the fuzzy logic controller.

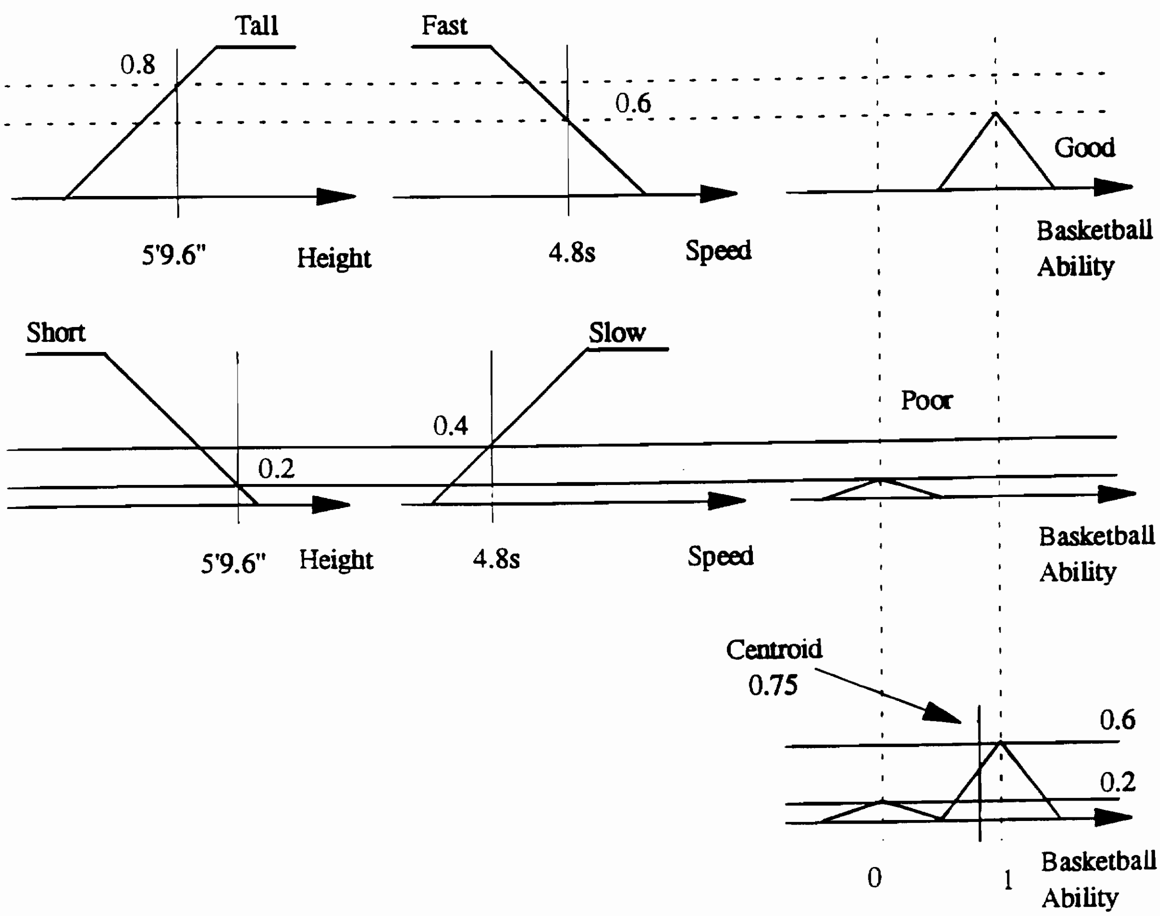


**Figure 3.3** Graphical illustration of fuzzy logic rules.

## 3.4 FUZZY LOGIC FORCE CONTROL IMPLEMENTATION

The force controller for surface exploration was implemented as a two input, single output fuzzy controller. The two inputs were force error and change in force error. The single output produced by the controller was used to adjust the Cartesian position of the end-effector. Three different sets of seven linguistic variables were used to describe the two inputs and single output. Force error was described as positive or negative, large, medium, small or zero (NL, NM, NS, ZE, PS, PM, PL). Similarly, the change in force error was described as increasing or decreasing, fast, moderate, slow or zero (DF, DM, DS, ZE, IS, IM, IF). The output, desired change in the Cartesian position of the end-effector, was described as approach or pull back, fast, moderate, slow or zero (BF, BM, BS, ZE, AS, AM, AF). These membership functions are shown in Figure 3.4.

The force controller rule base was generated from the following five general rules.

1) If the force error is small, then do nothing.

2) If the force error is positive large and increasing, then pull back fast.

3) If the force error is positive large and decreasing fast, then pull back slowly.

4) If the force error is negative large and decreasing, then approach fast.

5) If the force error is negative large and increasing fast, then do nothing.

The first rule occupies approximately 21 elements of the rule matrix. Rules two and four occupy an additional three elements each and rules three and five each fill one element. The 20 remaining elements of the rule matrix were assigned consequences to smooth the transitions between the specified rules. The final rule matrix is shown in Table 3.3.

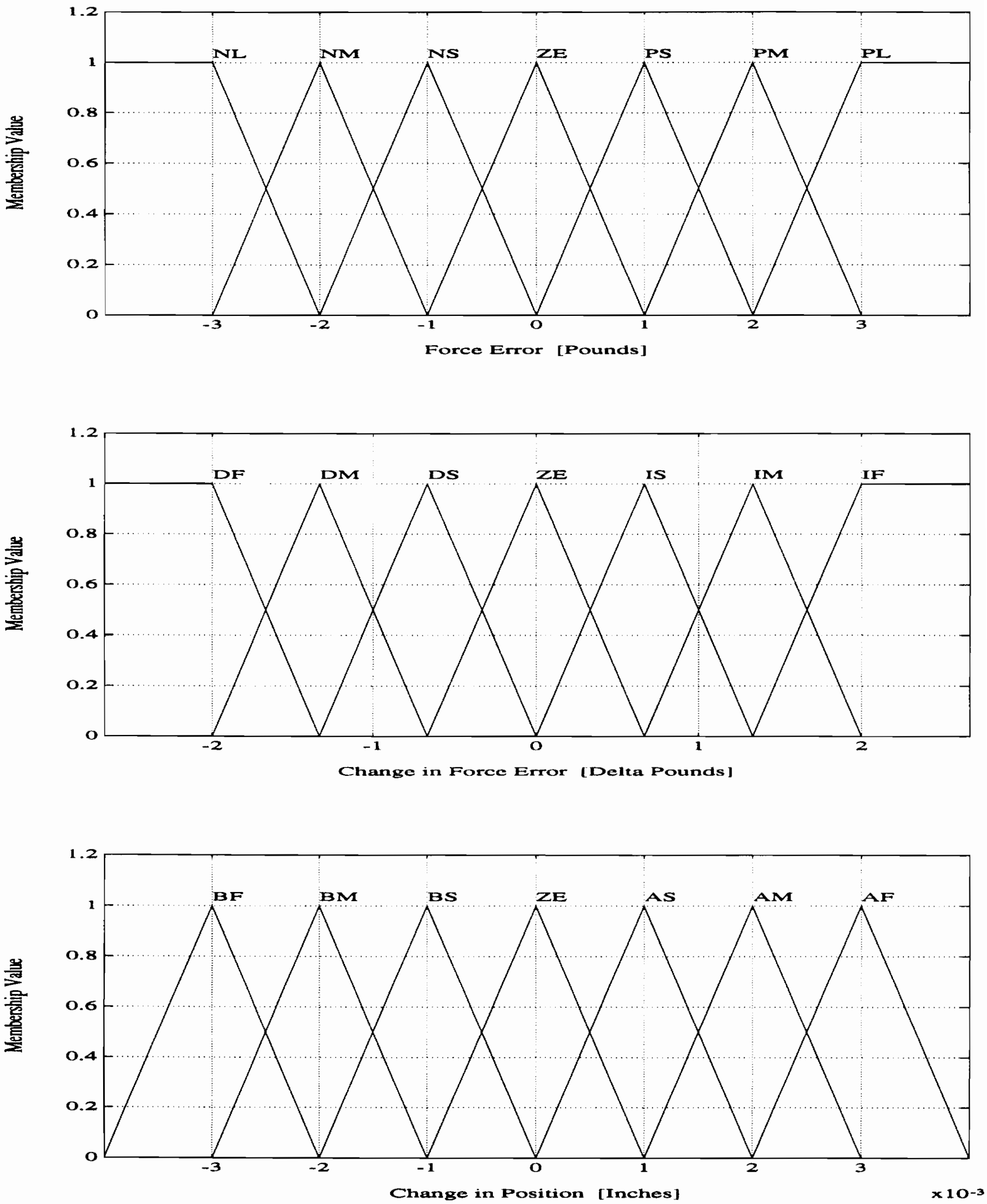


**Figure 3.4** Fuzzy logic force controller membership functions.

A three dimensional projection of the resulting control surface is shown in Figure 3.5. The hill above the zero planerepresents the controller instructing the robot to pull back. Similarly, the trench below the zero plane represents the controller instructing the robot to approach the object.

**Table 3.3** Fuzzy logic force controller rule matrix.

| | Change in Force Error | | | | | | |
|---|---|---|---|---|---|---|---|
| Force Error | DF | DM | DS | ZE | IS | IM | IF |
| PL | BS | BM | BF | BF | BF | BF | BF |
| PM | ZE | BS | BM | BM | BM | BM | BM |
| PS | ZE | ZE | ZE | ZE | ZE | ZE | BS |
| ZE | ZE | ZE | ZE | ZE | ZE | ZE | ZE |
| NS | AS | ZE | ZE | ZE | ZE | ZE | ZE |
| NM | AM | AM | AM | AM | AS | ZE | ZE |
| NL | AF | AF | AF | AF | AM | AS | ZE |

Examining the control surface reveals that the force controller's behavior is similar to that of a PD controller. The major difference is a dead-band across the change in force error axis for small force errors. This is a result of the first rule and consequently the requirement that the controller only needs to maintain a reasonable contact force. When this requirement is translated to the control surface, by the fuzzy logic controller, it results in a relatively flat response for small force errors irrespective of the change in the force error.

After the rule matrix has been selected, the membership functions for the inputs and output can be manipulated to fine tune the controller's response. Figure 3.1 shows the resulting contact force as the end-effector makes contact with a table top and then rolls along a spiral trajectory on its surface. Unfortunately the force oscillates, but it remains within ±1 pound of the desired 8 to 12 pound range. The inability of the controller to eliminate the oscillations can most likely be attributed to the design of the end-effector's compliance mechanism. This mechanism works best when it perpendicular to the surface and the applied force is along its sliding axis. Side forces due to the robot following an exploration trajectory tend to jam the mechanism or cause it to momentarily stick in place. This action results in impulse forces which disturb the force controller. The oscillations can arise because of the time delay between when the force/torque sensor transmits it's data to the controller and when the axis controllers are able to change the motion of the arm. Therefore, decreasing the speed at which the robot explores a surface and decreasing the force controller's gain will help dampen the oscillations. Unfortunately this will drastically increase the time it takes to explore an object's surface. Thus, a trade-off exists between force stability and exploration time.

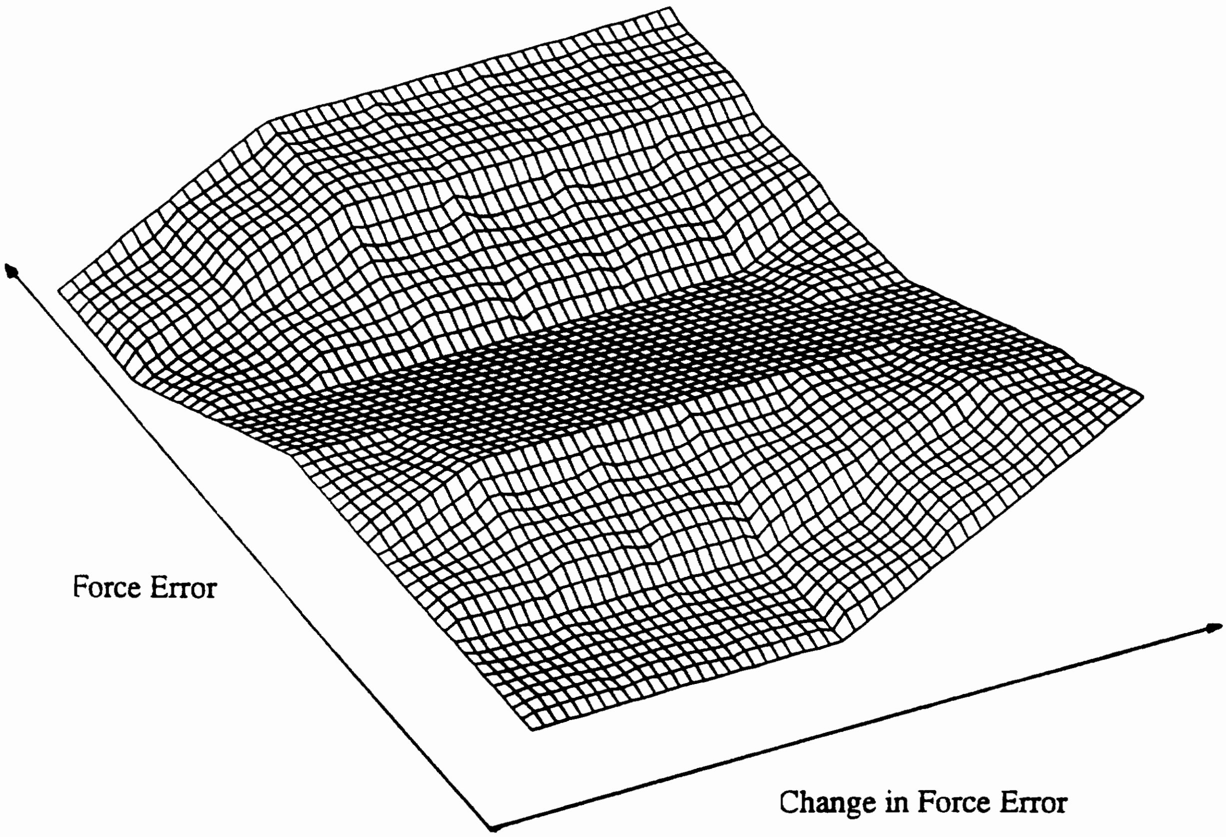


**Figure 3.5** Fuzzy logic control surface.

# 4.0 SHAPE ESTIMATION

The goal of the autonomous exploration algorithm is to determine the shape of an object as quickly and efficiently as possible. A geometric model will be used to mathematically describe the surface of the object. The model will have the implicit form $S(X, P) = 0$, where $X$ is a vector of inertial coordinates and $P$ is a vector of constant geometric parameters that fully describe the model. The components of the parameter vector, $P$, also determine the shape of the object.

The task of generating a description of an object is equivalent to estimating the geometric parameter vector. Because a sequence of surface measurements will be generated by the tactile probe as it travels over the object, a Kalman filter can be used to iteratively update an estimate of the parameter vector.

The motion of the probe will be guided by an optimal exploration algorithm. The algorithm will minimize a cost function that will cause the estimate of the parameter vector to converge as rapidly as possible. The function will be minimized by supplying the surface tracker with optimal search directions. The resulting trajectory on the object's surface will cause the estimate to converge faster than for any other trajectory.

The next section of this chapter will discuss the surface model used to describe the object's shape followed by a section on the statistical filter used to generate an estimate of the model's parameter vector. Autonomous exploration will be discussed in the third section. This will include the formalization of the cost function and its minimization. Experimental results will be presented in the fourth section.

The object's local radius of curvature can also be determined from the surface measurements. This will be the subject of the final section. Surface curvature will be reviewed followed by the development of a method to determine a surface's local radius of curvature. The section will be concluded with experimental results.

## 4.1 SHAPE REPRESENTATION

A shape can be approximated by an interconnected mesh of surface patches or it can be represented by an implicit geometric model. Surface patches are frequently used in computer graphics and occasionally in computer vision to approximate curved objects. An arbitrary surface approximation can be constructed from a variety of patches such as planar, Bézier, B-spline and Coons patches [31]. Implicit geometric models on the other hand can accurately represent a large portion of curved objects. Multiple implicit models such as quadrics and superquadrics can be pieced together to approximate more complex surfaces. For example Pentland [34] [35] constructs a very realistic desk chair by "Or-ing" together appropriately deformed superquadrics. He also constructs a natural scene of a person, a tree, and a rock by combining 56 deformed superquadrics and fractal surfaces. Barr [4] is able to create an industrial part along with nuts and bolts through Boolean combinations of superquadrics.

A tactile exploration system only gathers information at a limited number of points on the object. This would make implicit geometric models more desirable for shape representation because they automatically fill in the gaps between the measured points.

If surface patches were used, either the tactile probe would have to examine a majority of the object's surface or an algorithm would be needed to fill in the missing patches (most likely based on implicit models). Surface patches would be better suited to vision systems that gather global information about the object.

Many industrial parts can be described by superquadrics. However, they contain several nonlinear terms that require an extended Kalman filter estimator [43]. Removing the nonlinear terms from a superquadric reduces the general equation to that of a quadric. Because quadrics can be expressed as a linear combination of their parameters, a linear Kalman filter estimator can be used.

The only objects to be estimated in this research will be spheres, although curvature experiments will be conducted on other types of objects. The estimates will be restricted because it is much easier to determine the true parameters of a sphere than a general superquadric and spheres are easier to obtain. However, the autonomous exploration algorithm described in this chapter can be applied to either form.

The spheres will be represented as quadrics and their surface model will have the implicit form $S(X, P) = 0$. A general quadric that can be expressed in this form is an ellipsoid. Its implicit equation is:

$$X^T E X + 2F^T X - 1 = 0 \tag{1}$$

where

$$E = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}, \qquad F = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad and \qquad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2}$$

The parameter vector for equation (1) is $P = [a\ b\ c\ d\ e\ f\ u\ v\ w]^T$. This vector represents the length, rotation and translation of each of the three major axes of the ellipsoid. The formulas used to extract this information from the parameter vector are derived in Appendix A.

## 4.2 SHAPE ESTIMATOR

It is the shape estimator's job to determine values for the geometric parameter vector such that the ellipsoid they describe match the object being explored with a minimum amount of error. To this end, a linear Kalman filter estimator was used to minimize the mean squared error.

The individual components of the parameter vector can not be directly measured, but an indirect measurement can be obtained by placing the end-effector in contact with the surface. The sensor fusion capability of the Kalman filter can then be exploited to combine the unrelated data from several sensors into a single measurement. The general formulation of this process for an arbitrary number of sensors was presented in [6]. The following discussion will be specific and limited to contact position and local surface normal.

The joint-mounted encoders along with a knowledge of the manipulator's kinematics were used to determine the contact location. The local surface normal at the point of contact was determined from the contact forces measured by the force/torque sensor. Both the contact location and local surface normal were transformed to the inertial coordinate frame and then formulated as linear measurements of the parameter vector.

Substituting the contact location into the surface model (equation (1)) results in the measurement equation $AP = 1$ where

$$A = [x^2\ 2xy\ 2xz\ y^2\ 2yz\ z^2\ 2x\ 2y\ 2z].$$

Formulating a linear measurement from the local surface normal is a bit more involved. To begin, let $N$ be a unit vector in the normal direction. In terms of the surface model, $S(X, P)$, $N$ can be written as:

$$N = \frac{\frac{\partial S(X,P)}{\partial X}}{\left\|\frac{\partial S(X,P)}{\partial X}\right\|} = \frac{EX + F}{\|EX + F\|} = \frac{BP}{\|BP\|}$$

where

$$B = \begin{bmatrix} x & y & z & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & x & 0 & y & z & 0 & 0 & 1 & 0 \\ 0 & 0 & x & 0 & y & z & 0 & 0 & 1 \end{bmatrix}.$$

Unfortunately, the normalization term, $\|BP\|$, is nonlinear. However, if the normal direction is broken down into its three components, it can be written as:

$$N = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = \frac{1}{\|BP\|} \begin{bmatrix} b_1P \\ b_2P \\ b_3P \end{bmatrix}$$

where $b_i$ is the *ith* row of $B$. The equations

$$n_1 = \frac{b_1P}{\|BP\|}, \qquad n_2 = \frac{b_2P}{\|BP\|}, \ and \qquad n_3 = \frac{b_3P}{\|BP\|}$$

are still nonlinear in the parameter vector, but they can be rearranged as:

$$\frac{n_1}{n_2} = \frac{b_1P}{b_2P}, \qquad \frac{n_2}{n_3} = \frac{b_2P}{b_3P}, \ and \qquad \frac{n_3}{n_1} = \frac{b_3P}{b_1P}$$

or

$$n_1b_2P - n_2b_1P = 0$$

$$n_2b_3P - n_3b_2P = 0$$

$$n_3b_1P - n_1b_3P = 0.$$

The last three equations are linear measurements of the parameter vector. In matrix form they are $GP = 0$ where

$$G = \begin{bmatrix} n_2x & n_2y-n_1x & n_2z & -n_1y & -n_1z & 0 & n_2 & -n_1 & 0 \\ 0 & n_3x & -n_2x & n_3y & n_3z-n_2y & -n_2z & 0 & n_3 & -n_2 \\ -n_3x & -n_3y & n_1x-n_3z & 0 & n_1y & n_1z & -n_3 & 0 & n_1 \end{bmatrix}.$$

If the error in the sensor data is modeled as a zero mean Gaussian noise vector $\nu$ with covariance matrix $R = E\{\nu\nu^T\}$, then the actual surface measurement can be written as $Z(X, P) = HP + \nu$ where

$$H = \begin{bmatrix} A \\ G \end{bmatrix}.$$

This allows an estimate of the parameter vector, $\hat{P}$, to be iteratively generated with a Kalman filter estimator. The update equations for the estimator are:

$$\hat{P}_k = \hat{P}_{k-1} + Q_{k-1} H_k^T R^{-1} [Z_k - \hat{Z}_k]$$

$$Q_k = [Q_{k-1}^{-1} + H_k^T R^{-1} H_k]^{-1} \qquad (3)$$

where $Z = [1\ 0\ 0\ 0]^T$ is the actual measurement without noise, $\hat{Z}_k = H_k P_{k-1}$ is the estimated measurement for the current sensor data and $Q = E\{[P - \hat{P}][P - \hat{P}]^T\}$ is the estimate's covariance matrix.

## 4.3 OPTIMAL EXPLORATION

As the Kalman filter updates the estimate based on surface measurements, the covariance matrix, $Q$, is also updated. Initially the filter places very little confidence in its estimate, electing to heavily weight the incoming measurements when computing the estimate updates even though the measurements are corrupted with noise. As more measurements are incorporated into the estimate, the magnitude of the covariance matrix is reduced to reflect the increased confidence in the estimate. This causes the filter to de-emphasize the noisy measurements which in turn reduces any changes to the estimate. In theory, enough measurements could be taken to reduce the uncertainty of the estimate to zero, in which case the estimate would be unaffected by further measurements. This condition is possible as long as the true surface parameters being estimated are constant and the measurements are rich enough in information to allow the parameters to be accurately estimated. Repeated measurements of a fixed point on the object will obviously not be rich enough. Neither will measurements on a great circle of a sphere. Based on the second set of measurements it would be impossible to distinguish a sphere from a cylinder, or for that matter, an ellipsoid with two axes of equal length.

Examining the covariance update equation (3) for the Kalman filter reveals that the updates are a function of the measurements which in turn depend on the contact location and surface normal. Therefore, the covariance not only represents the uncertainty of the estimate but is also a function of the contact location. This indicates that it should be possible to determine a direction to move the contact point that will maximize the rate of reduction of the estimate's covariance matrix [6].

A search direction that optimizes the estimate's rate of convergence will be derived by first formulating a scalar cost function associated with the change in the covariance matrix and then minimizing it over all direction vectors tangent to the surface at the current contact location. To construct a cost function, a scalar measure of the estimates uncertainty is needed. This can be obtained by taking the trace of the covariance matrix. Initially this value will be large, but as the uncertainty in the estimate is removed this value will go to zero. The change in uncertainty from one step to the next can be obtained by the difference in the trace of the respective covariance matrices. This method was first presented in [6]. Using equation (3), the cost function, $J(Q)$, can be written as:

$$
\begin{aligned}
J(Q_k) &= -\mathrm{trace}(Q_k^{-1} - Q_{k-1}^{-1}) \\
&= -\mathrm{trace}(H_k^T R^{-1} H_k).
\end{aligned}
$$

The extrema of this cost function can be found by computing its partial derivative with respect to $X$ and setting it equal to zero:

$$
\begin{aligned}
\frac{\partial J}{\partial X} &= -\frac{\partial}{\partial X} trace(H^T R^{-1} H) \\
&= -\frac{\partial}{\partial H}[trace(H^T R^{-1} H)]\frac{\partial H^T}{\partial X} \\
&= -2H^T R^{-1}\frac{\partial H^T}{\partial X} = 0.
\end{aligned}
\tag{4}
$$

The optimal search direction can be found by evaluating equation (4) for all direction vectors on the tangent plane of the surface at the current contact location.

The search directions can be generated with:

$$
dX = M\begin{bmatrix}\cos(\rho) \\ \sin(\rho)\end{bmatrix} \qquad 0 \leq \rho < 2\pi
$$

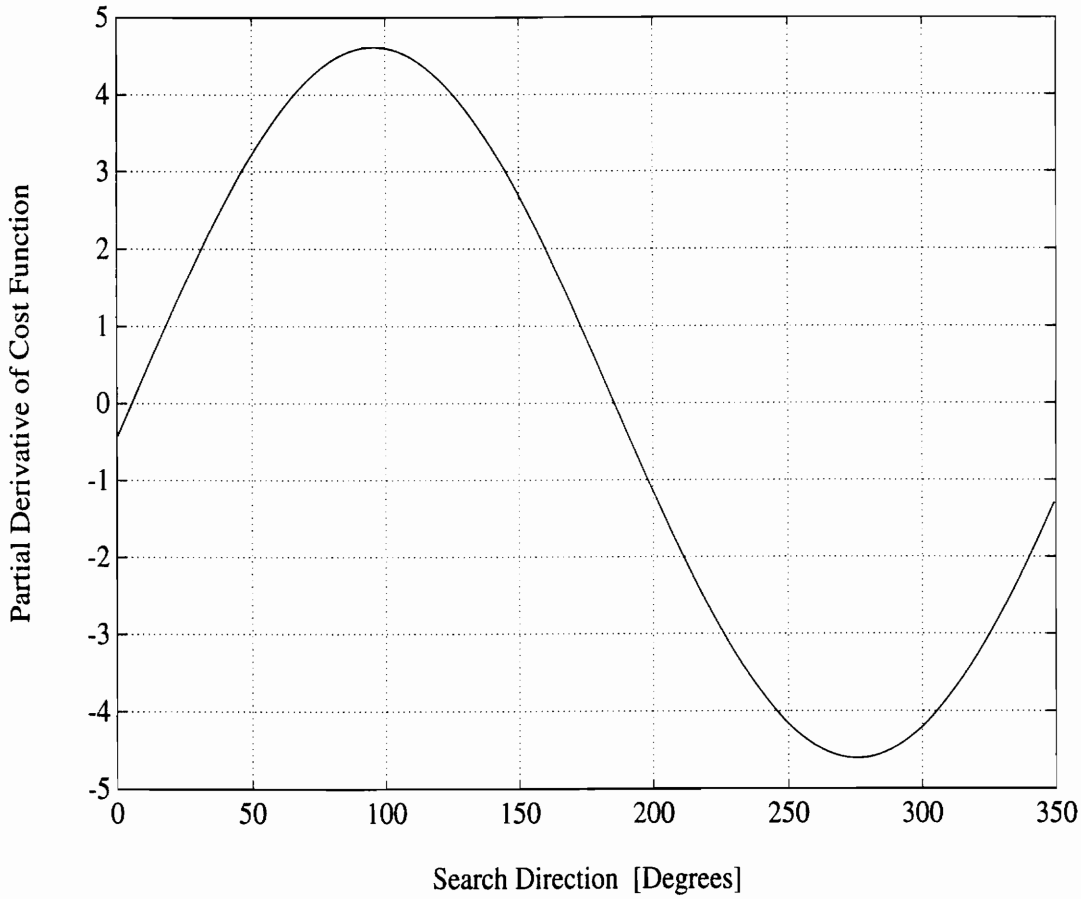


**Figure 4.1** Partial derivative of cost function for all search directions on the tangent plane.

where $M$ is a linear transformation that projects the X-Y plane onto the local tangent plane of the surface. A suitable transformation is:

$$M = \left[ [N \times dX_{prev}] \times N \quad N \times dX_{prev} \quad N \right]$$

where $dX_{prev}$ is the previous search direction and $\times$ is the cross product operator.

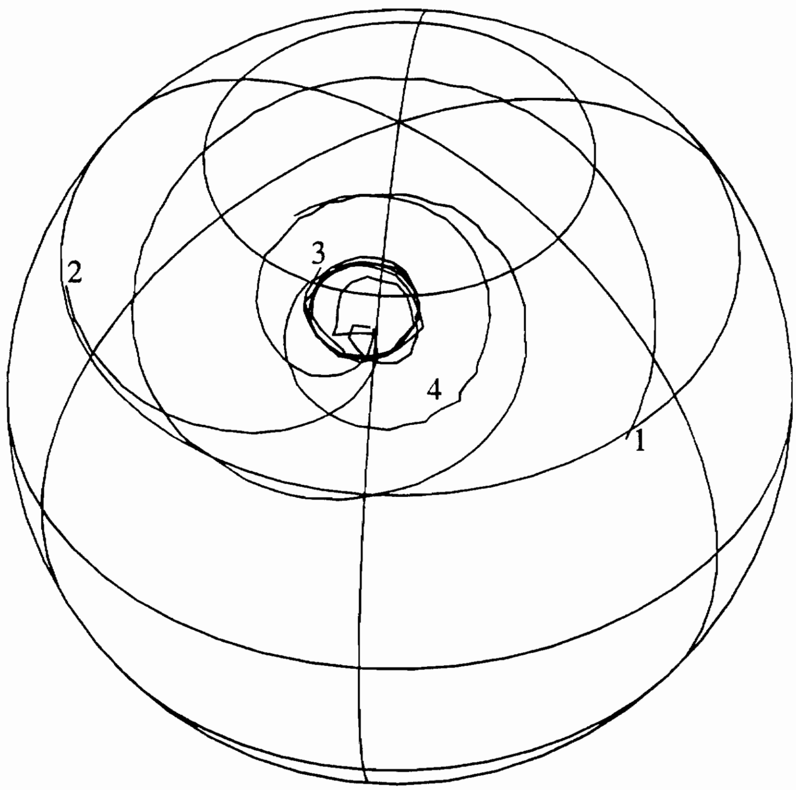


**Figure 4.2** Exploration trajectory on 3½ inch radius sphere.

When searching for the extrema of the cost function, a minimum can be distinguished from a maximum by examining the sign of the slope of equation (4) at its zero crossings. If the tangent plane is searched in a counter clockwise direction ($\rho$ increasing from $0 \rightarrow 2\pi$), the value of equation (4) will be increasing when the cost function is at its minimum (see Figure 4.1.)

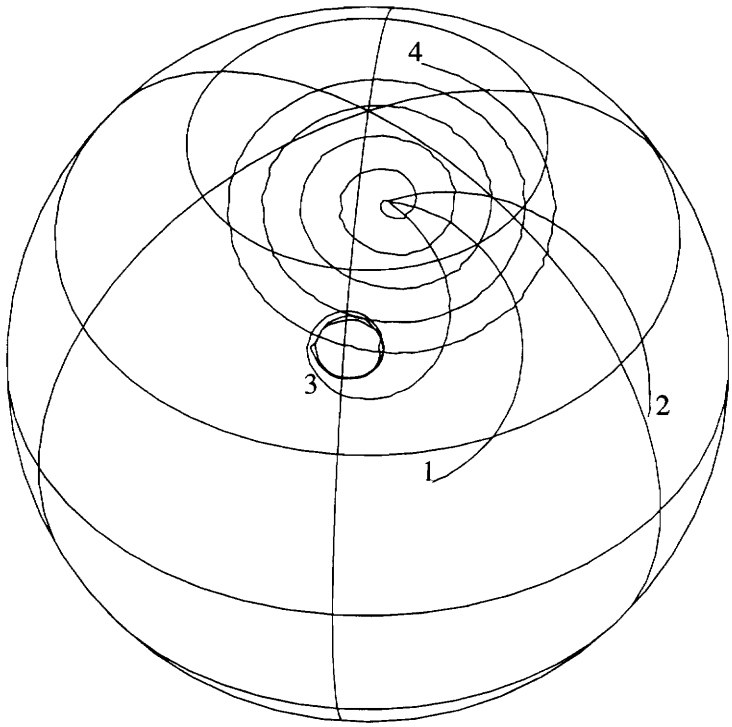


**Figure 4.3** First set of exploration trajectories on 9 inch radius sphere.

## 4.4 Exploration Results

As mentioned earlier, shape estimation experiments were conducted on spherical objects. The objects were two plexiglass hemispheres attached to a table top. The two hemispheres had radii of 3½ and 9 inches respectively.

After manually placing the end-effector near the object's surface, four paths were used to explore it. These paths are labeled 1 through 4 in Figures 4.2, 4.3 and 4.4. The first path was the optimal trajectory generated by the autonomous exploration algorithm. The second and third paths were shifted to the left and right of the optimal trajectory by

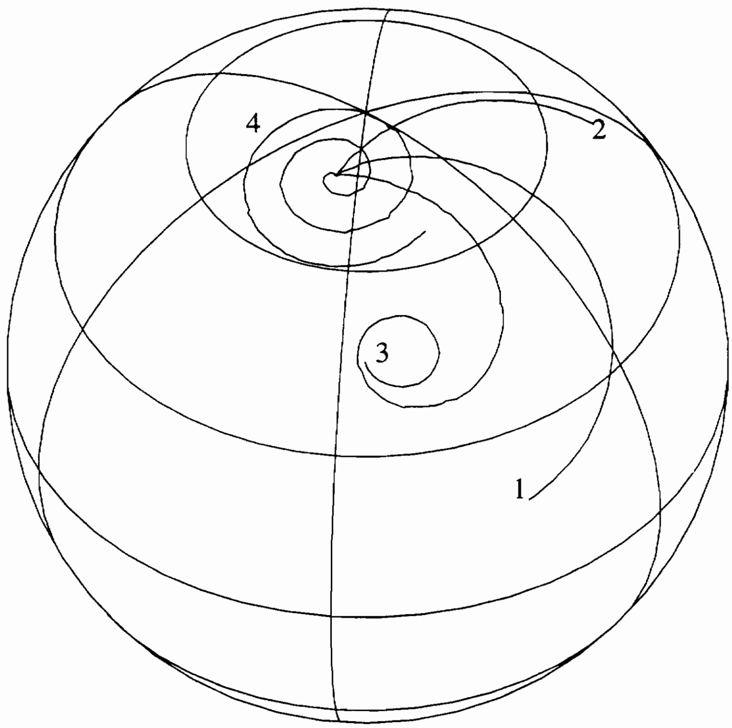


**Figure 4.4** Second set of exploration trajectories on 9 inch radius sphere.

±30° and the fourth path was a predetermined outward spiral.

The four exploration paths were used to investigate the effectiveness of the autonomous exploration algorithm. To compare the ellipsoidal estimates of the spheres generated along the different paths, the mean squared errors of the parameter vectors were computed. However, the parameter vectors had to be modified first.

In addition to the lengths of an ellipsoid's three major axes, a parameter vector also represents rotations and translations along these axes. Because the orientation of spheres is not unique, the rotation information in the parameter vector must be eliminated.

This can be accomplished be removing the cross product terms from the quadric equation (parameters $b$, $c$ and $e$ in equation (2)) with a change of basis operation (see Appendix A.) The resulting parameter vector, $P'$, is $[a'\ 0\ 0\ d'\ 0\ f'\ u'\ v'\ w']^T$. The parameters $a'$, $d'$ and $f'$ are functions of the sphere's radius and location in the inertial coordinate system. The three remaining non-zero parameters ($u'$, $v'$ and $w'$) are only dependent on the location of the sphere. These will be ignored when computing the error in the parameter vector because the locations of the spheres are only known approximately. While $a'$, $d'$ and $f'$ are functions of the sphere's location, they also depend on the sphere's radius which is known. After approximating the locations of the spheres and computing the near-true values of $a'$, $d'$ and $f'$, the shape estimates will be compared by computing the mean squared error between these three parameters and the modified parameter vectors.

The mean squared error will not be 100% accurate because the true values of $a'$, $d'$ and $f'$ cannot be computed exactly, but it will be accurate enough to compare the rates of convergence for the four estimates. The location of the spheres was approximated from the best estimate obtained by modeling the objects as actual spheres rather than general ellipsoids. A spherical model only requires four parameters instead of the nine needed to model an ellipsoid and thus provides a more accurate estimate.

The mean squared errors obtained from exploring a plexiglass hemisphere with a 3½ inch radius and a 9 inch radius are shown in Figures 4.5, 4.6 and 4.7 (Figure 4.6 and 4.7 are separate trials on the same hemisphere.) The figures compare the rates of convergence of the estimates generated from the four exploration trajectories shown in

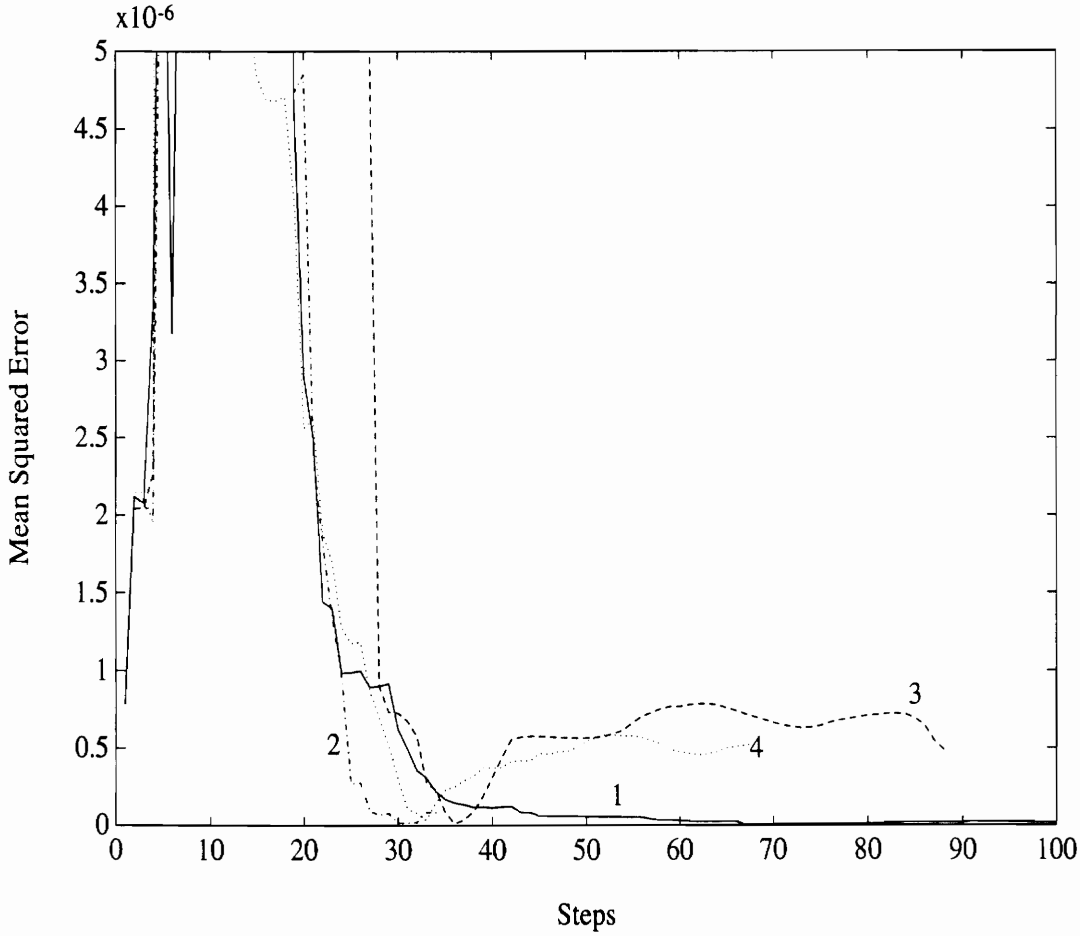


**Figure 4.5** Mean squared error for 3½ inch radius sphere.

Figures 4.2, 4.3 and 4.4. The results look fairly inconclusive. This could be due to the fact that a sphere is very uninteresting. It looks the same no matter which direction one looks when their vantage point is on its surface. The results might have been more conclusive if a more interesting (complex) object and shape model had been used.

Based on a visual interpretation of the mean squared error plots, the preprogrammed outward spiral (path 4) appears to be suboptimal. In two of the three sets of trajectories its performance is far worse than the three paths related to the

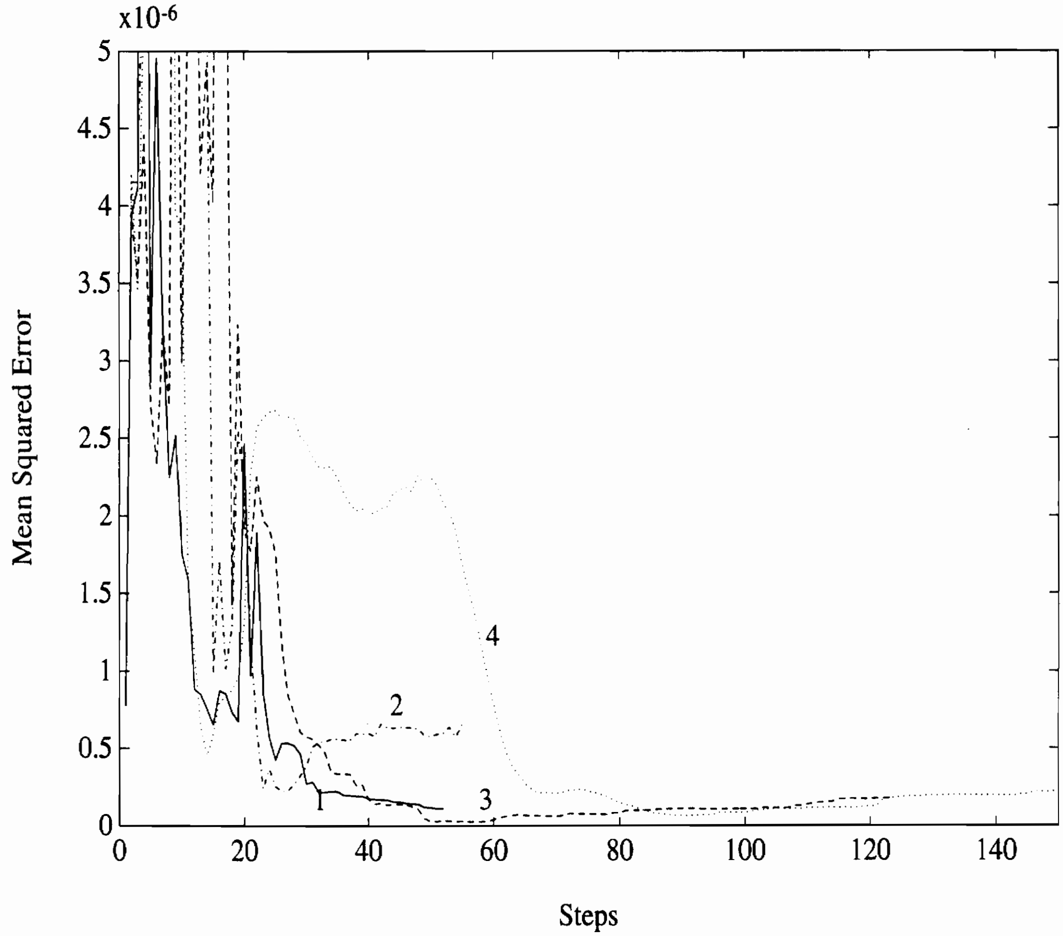


**Figure 4.6** Mean squared error for first trial on 9 inch radius sphere.

autonomously generated trajectory. In addition, the optimal trajectory appears to have the most consistent rate of convergence between the different objects.

In some instances, exploration had to be stopped prematurely. This was a result of the second axis of the manipulator's wrist being physically constrained to ±90° of rotation. The location of the sphere and the exploration trajectory on its surface would eventually place the manipulator in a configuration that would over flex the wrist. Some trajectories, such as the optimal trajectory on the 3½ inch sphere and the outward spiral

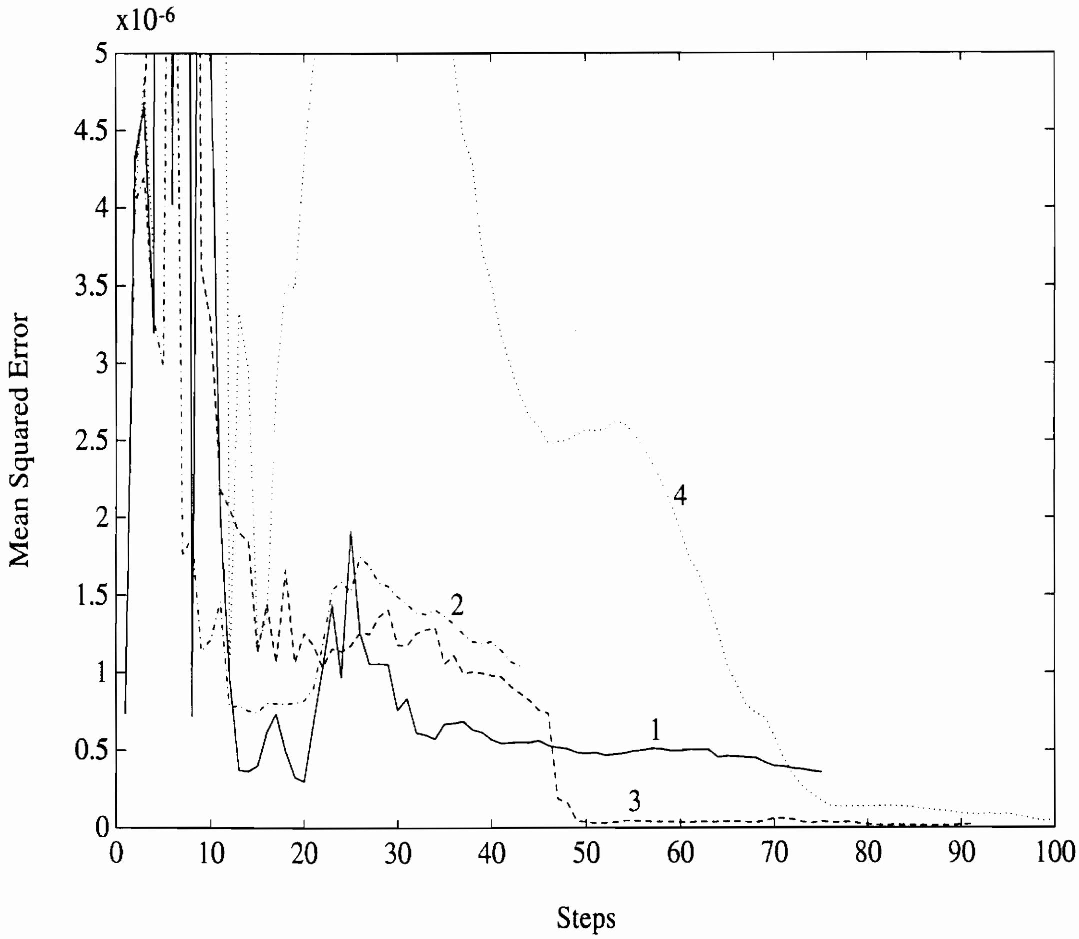


**Figure 4.7** Mean squared error for second trial on 9 inch radius sphere.

on the 9 inch sphere, would progress farther than others before this condition occurred.

Often more than 100 surface measurements were needed before the geometric parameters would begin to converge. Unfortunately the exploration trajectory often had to be aborted before this many measurements could be taken. Figures 4.8 and 4.9 show the estimated parameters for the first trial of optimal and spiral trajectories on the 9 inch sphere. The parameters had not yet converged for the optimal trajectory when it was stopped after 55 steps. Figure 4.9 shows that the parameters have converged after

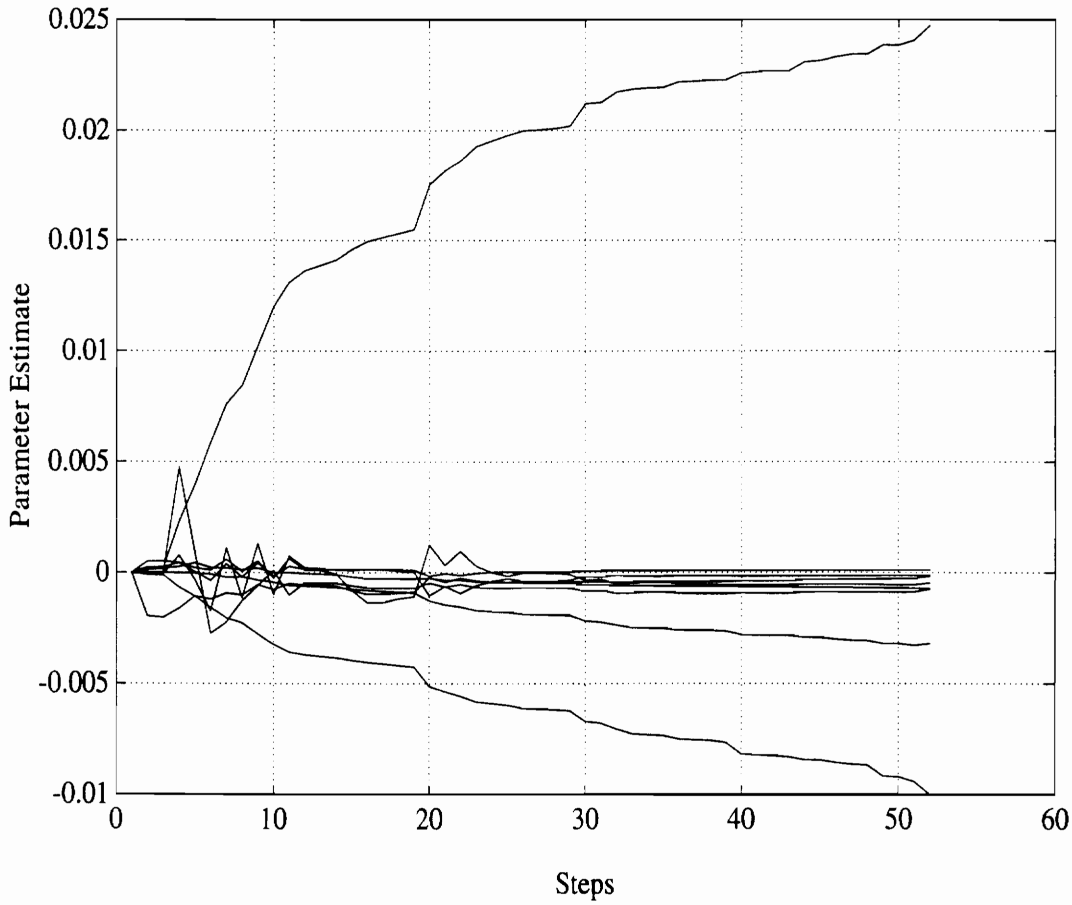


**Figure 4.8** Estimated parameters for first optimal trajectory on 9 inch radius sphere.

approximately 350 steps for the spiral trajectory.

The axis lengths and locations of the plexiglass hemispheres, computed from the parameter estimates of each path, are summarized in Table 4.1. It is clear that the longer trajectories give better estimates irrespective of the method used to generate the trajectory.

In summary, it is inconclusive that the autonomous exploration algorithm results in the fastest converging estimate. However, due to the physical constraints of the manipulator the optimal exploration trajectory does not always generate enough

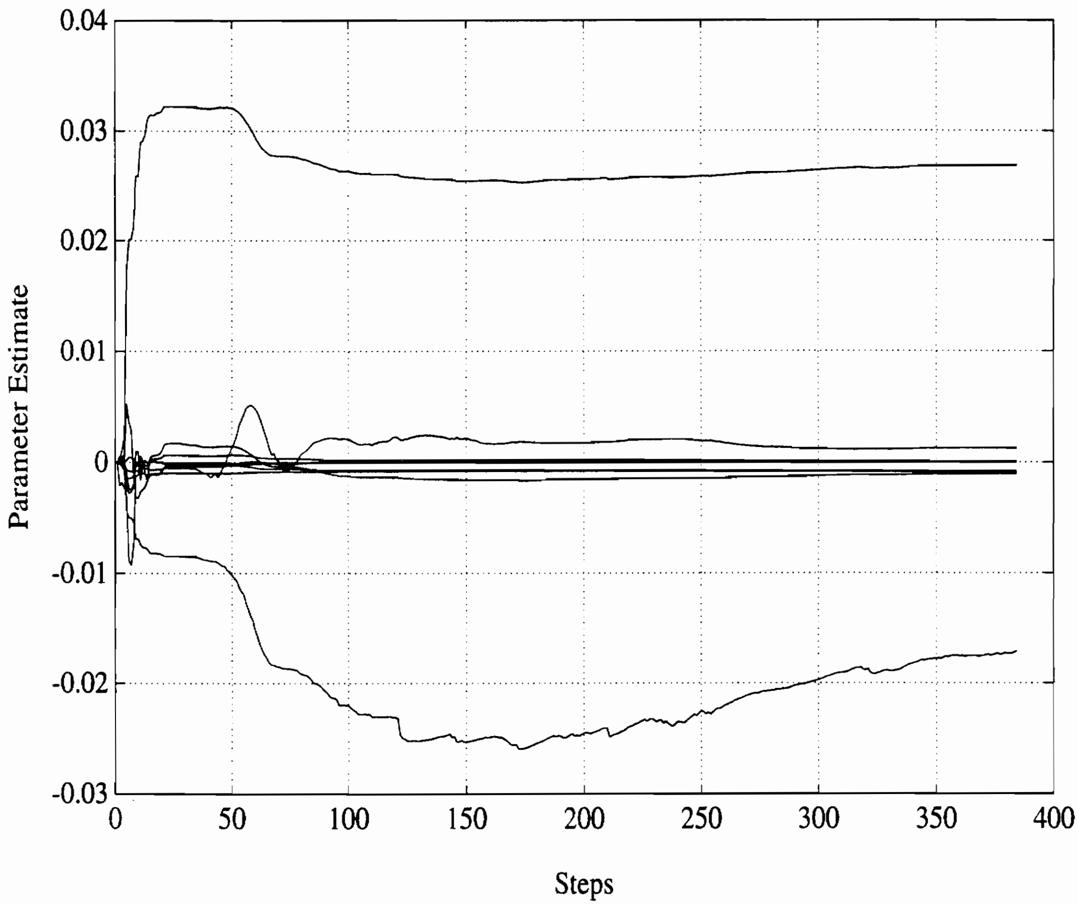


**Figure 4.9** Estimated parameters for first spiral trajectory on 9 inch sphere.

measurements to extract an accurate estimate of the object. Instead, a predetermined trajectory sometimes produces more surface measurements than the optimal trajectory resulting in a better estimate.

**Table 4.1** Estimated axis lengths and locations.

| Path | Steps | Axis Length (inches) | | | Location (inches) | | |
|---|---|---|---|---|---|---|---|
| | | X | Y | Z | X | Y | Z |
| 3½ Inch Sphere | | | | | | | |
| 1 | 133 | 5.77 | 5.67 | 5.25 | 29.21 | -0.89 | -16.45 |
| 2 | 33 | 7.95 | 6.90 | 10.46 | 29.00 | -3.40 | -20.29 |
| 3 | 87 | 1.09 | 1.32 | 0.29 | 26.20 | -0.41 | -12.65 |
| 4 | 67 | 1.98 | 2.40 | 0.51 | 26.24 | -0.49 | -12.84 |
| First Trial On 9 Inch Sphere | | | | | | | |
| 1 | 51 | 24.06 | 25.25 | 39.61 | 45.09 | 9.83 | -40.84 |
| 2 | 54 | 5.06 | 0.00 | 0.00 | 29.49 | -10.91 | -6.96 |
| 3 | 122 | 7.85 | 8.43 | 4.79 | 26.18 | 2.23 | -14.02 |
| 4 | 383 | 9.58 | 9.33 | 8.45 | 30.30 | 0.61 | -15.23 |
| Second Trial On 9 Inch Sphere | | | | | | | |
| 1 | 74 | 0.00 | 124.22 | 0.00 | 56.79 | -59.85 | 98.13 |
| 2 | 42 | 0.00 | 0.00 | 2.77 | 33.42 | -2.68 | -6.08 |
| 3 | 90 | 10.21 | 9.86 | 8.63 | 29.39 | 4.66 | -16.35 |
| 4 | 123 | 7.53 | 7.71 | 5.43 | 29.48 | 4.91 | -12.39 |

## 4.5 CURVATURE SENSING

Surface curvature will be described by first examining some properties of space curves and then extending them to surfaces and surface curves. A method for sensing the surface's local radius of curvature will then be developed.

A space curve can be described implicitly as $P(\boldsymbol{X}) = 0$ or parametrically as $P(u) = 0$. In the first form $\boldsymbol{X}$ is a coordinate vector and in the second form $u$ is a parameter of the curve such as time. Note that in this section vectors will be expressed in boldfaced italics and scalars in regular italics.

Three important properties of space curves are arc length, tangent direction and curvature [44, pp. 5-15]. Arc length is the distance measured on a curve. It is determined by

$$s(u) = \int_{u_0}^{u} \left( \frac{d\boldsymbol{X}}{du} \cdot \frac{d\boldsymbol{X}}{du} \right)^{1/2} du \ .$$

The tangent direction of a curve is defined as the rate of change of the coordinate vector, $d\boldsymbol{X}$, along the curve with respect to the rate of change of the arc length, $ds$. Its equation is

$$\boldsymbol{t} = \frac{d\boldsymbol{X}}{ds} \ .$$

Finally, the rate of change of the tangent with respect to the rate of change of the arc length is curvature. It is given by

$$\boldsymbol{k} = \frac{d\boldsymbol{t}}{ds} = k\boldsymbol{n}$$

where $\boldsymbol{n}$ is the principle normal of the curve. The radius of curvature at a point on the line is given by $R = k^{-1}$.

Surfaces can also be described implicitly as $S(\boldsymbol{X}) = 0$ or parametrically as $S(\boldsymbol{u}, v) = 0$. The tangent plane of the surface is defined at a point by the plane with direction vectors $\boldsymbol{X}_u$ and $\boldsymbol{X}_v$ [44, p. 62]. The surface normal is orthogonal to this plane so

$$N = \frac{\boldsymbol{X}_u \times \boldsymbol{X}_v}{\| \boldsymbol{X}_u \times \boldsymbol{X}_v \|}$$

where

$$\boldsymbol{X}_u = \frac{\partial \boldsymbol{X}}{\partial u} \qquad \mathit{and} \qquad \boldsymbol{X}_v = \frac{\partial \boldsymbol{X}}{\partial v} .$$

When a curve is projected onto a surface its curvature vector can be decomposed into two components: $\boldsymbol{k} = \boldsymbol{k}_n + \boldsymbol{k}_g$ [44, pp. 73-77]. The vector $\boldsymbol{k}_n = \boldsymbol{k} \cdot \boldsymbol{N}$ is the normal curvature vector and $\boldsymbol{k}_g$ is the tangential curvature vector or geodesic curvature vector. The geodesic curvature vector is strictly a property of the surface curve. However, the normal curvature vector is a property of the surface and is direction dependent *i.e.* curves passing through the same point on the surface with different tangents could yield different values of normal curvature. The directions for which normal curvature are maximized and minimized are called the principle curvature directions. These directions are always orthogonal except at singular points called umbilics [44, p. 80]. Normal curvature is constant for all directions at these points. For example, there are four umbilics on an ellipsoid with unequal axes and all points on spheres and planes are umbilics.

Enough information is available when exploring an object to determine its local radius of curvature (the inverse of normal curvature) along the exploration trajectory. The two critical pieces of information needed are the rate of change of the trajectories tangent vector projected onto the surface normal, $dt{\cdot}N$, and the rate of change of the arc length, $ds$. The quantity $dt{\cdot}N$ can be approximated by the incremental change in the surface normal, $\Delta N$.

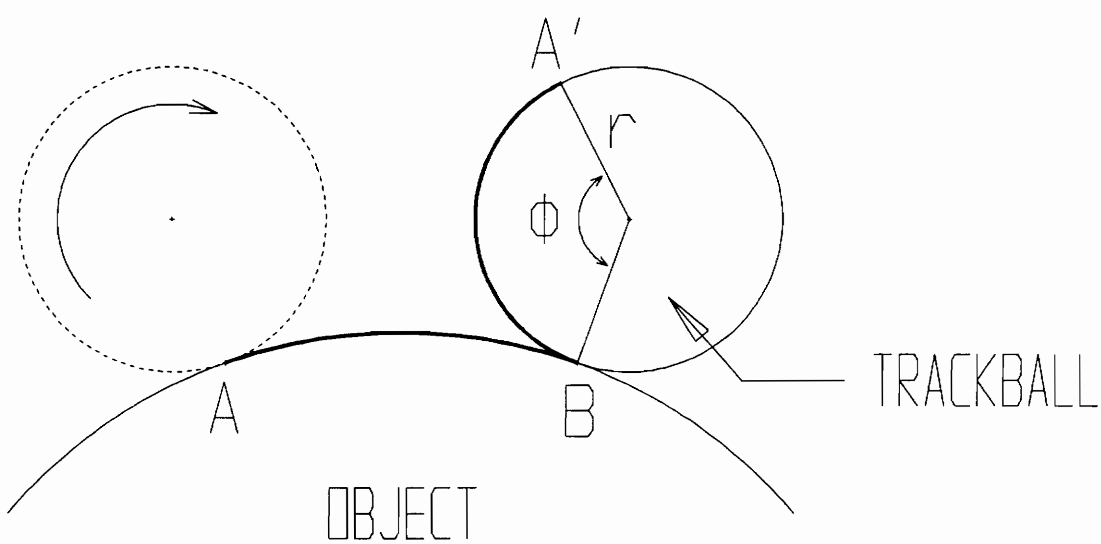


**Figure 4.10** Illustration of how to compute arc length traversed on an object's surface.

Arc length can be approximated with the trackball. Encoders mounted inside the trackball housing measure the rotation of the ball. This can be used in conjunction with the known radius of the ball to compute the traversed arc length on its surface. The arc length traversed on the object's surface is equal to the arc length on the ball. This is illustrated in Figure 4.10. The quantity $ds$ can now be approximated from the incremental arc length, $\Delta s$. The normal radius of curvature can be determined by

$$R_n = \frac{1}{k_n} \approx \frac{\Delta s}{\Delta N} \ . \tag{5}$$

In an attempt to filter the data when determining the local radius of curvature, a moving average was computed. This was accomplished by averaging two adjacent surface measurements. The incremental values of $\Delta N$ and $\Delta s$ were then computed from the difference between current, averaged, surface measurement and the previous, averaged, measurement. The local radius of curvature was then computed from equation (5).

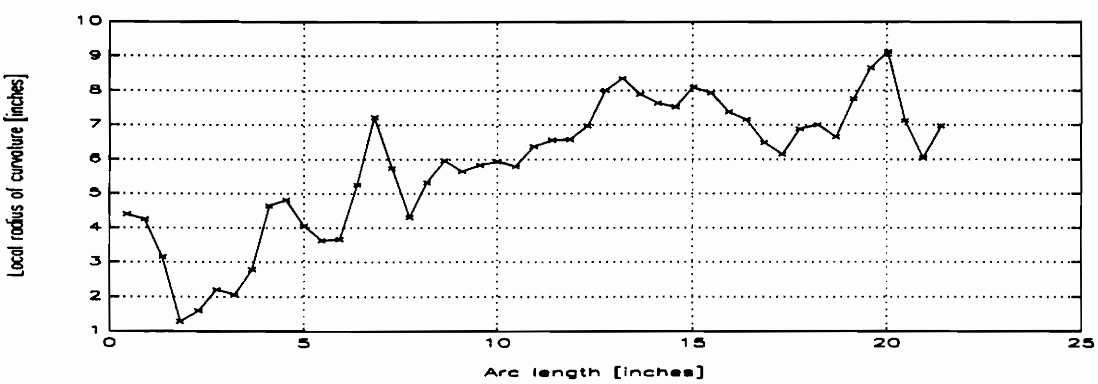


**Figure 4.11** Local radius of curvature measured on a 9 inch radius sphere.

The accuracy of the previously described method was examined by computing the local radius of curvature along the optimal exploration trajectories of the previous section. Figure 4.11 shows the results for a sphere with a 9 inch radius. It appears that the trackball slid along the surface instead of rolling at the beginning of the trajectory. This

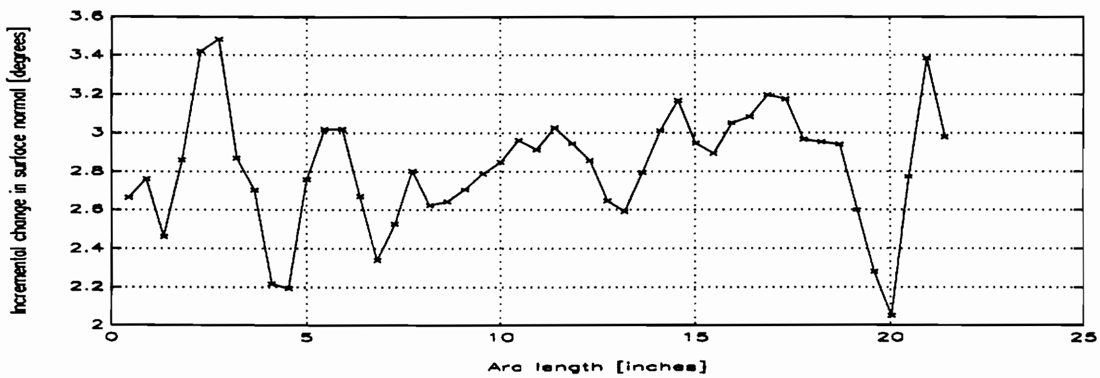


**Figure 4.12** The incremental change in surface normal is relatively constant.

can be verified by noting that the incremental change in surface normal is relatively constant, shown in Figure 4.12, but the incremental arc length in Figure 4.13 increases as the exploration progresses. It is quite likely that the trackball slipped because the plexiglass spheres are very smooth and the trackball's surface is highly polished.

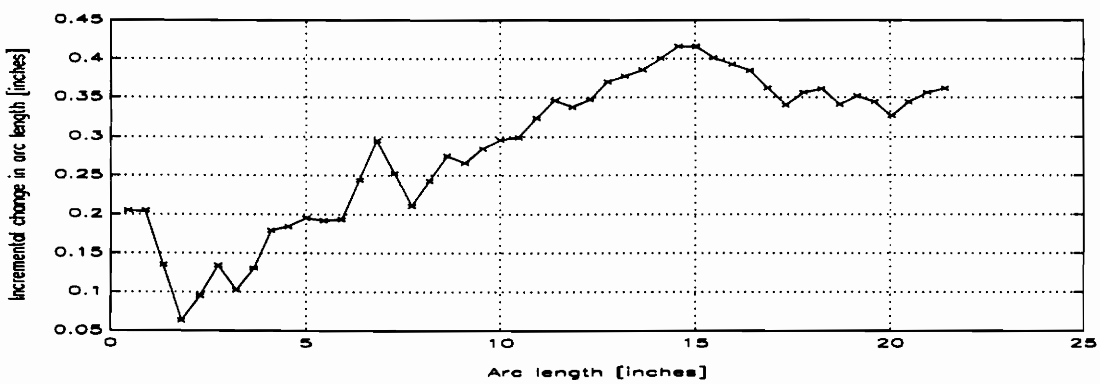


**Figure 4.13** The incremental change in arc length indicates the trackball slipped at the beginning of the exploration trajectory.

The 3½ inch sphere gave slightly better results. The radius of curvature along the optimal exploration trajectory is shown in Figure 4.14.

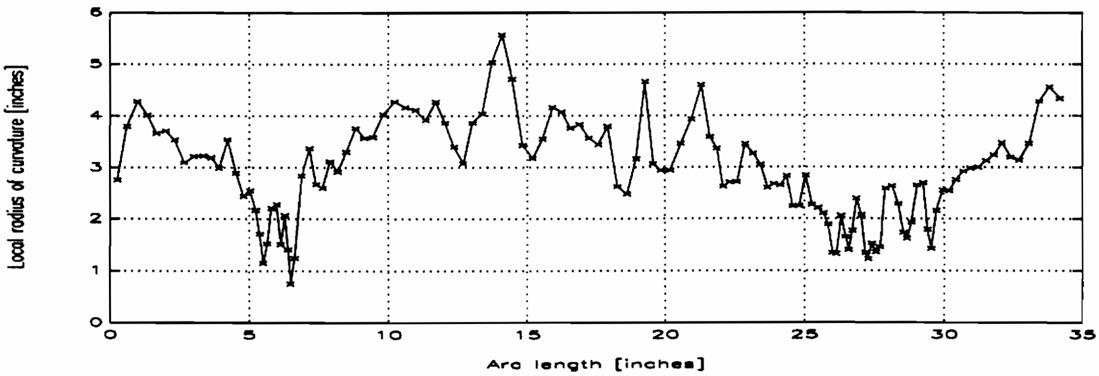


**Figure 4.14** Local radius of curvature measured on 3½ inch radius sphere.

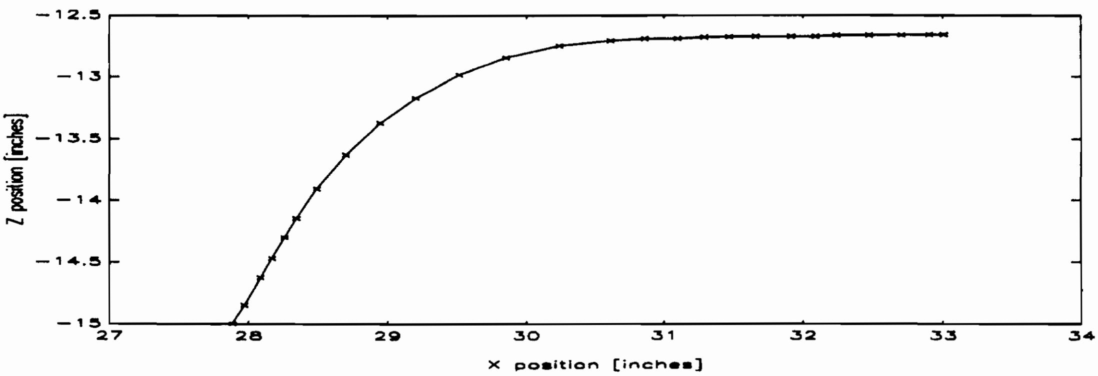


**Figure 4.15** Exploration trajectory on counter weight. The trajectory begins at the right and progresses to the left.

It should be possible to identify the edge of piece-wise planar objects by looking for discontinuities in their surface curvature. A lead counter weight with two planar faces meeting at a 30° angle was used to verify this. A side view of the exploration trajectory on the counter weight is shown in Figure 4.15 and the computed radius of curvature is shown in Figure 4.16. Ideally the radius of curvature should be infinite along the flat faces and momentarily drop to zero at the edge. However, the counter weight had a

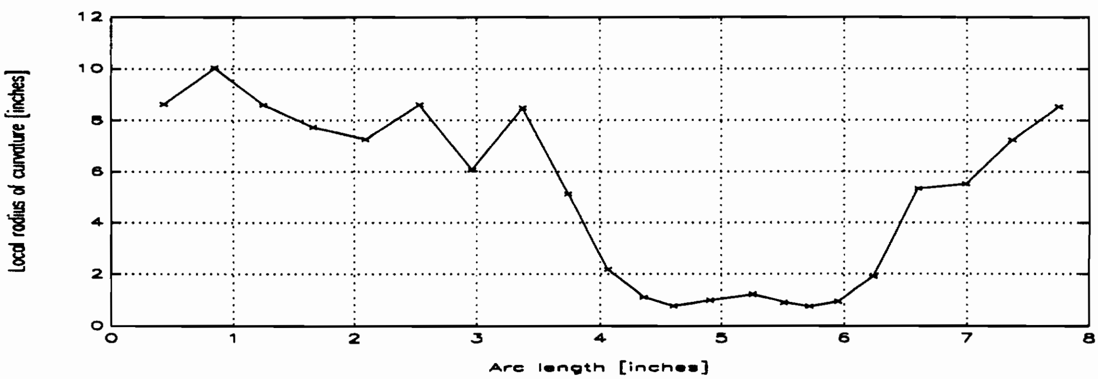


**Figure 4.16** Local radius of curvature measured on counter weight.

rounded edge and a rough surface texture. The rounded edge prevented the radius of curvature from reaching zero and the rough texture caused the measured surface normal to fluctuate. These fluctuations resulted in a finite radius of curvature for the flat faces. In spite of these imperfection the object's edge can be identified. However, due to measurement errors in the surface normal, contact location, and arc length, the location can only be determined with ±1 inch accuracy.

## 5.0 CONCLUSION

A shape exploration and estimation system has been implemented on a MERLIN 6540 Industrial Robot. This system consists of a fuzzy logic force controller, a Kalman filter quadric surface estimator, an optimal on-line trajectory generator, and a surface curvature measurement routine.

A fuzzy logic controller was deemed the most appropriate method of maintaining contact with a hard surface and regulating the applied force. This is because nonlinear control with general or nonspecific constraints can easily be implemented through fuzzy logic. The results of surface exploration experiments revealed that the controller performed adequately to maintain contact but exhibited significant oscillations. These oscillations can be attributed to the nonlinear effect of friction between the trackball and its housing, uncertainty in the kinematic equations, and the stepper motor construction of the MERLIN robot.

The Kalman filter quadric surface estimator performed well during computer simulations conducted prior to this thesis. Unfortunately the experimental results were less impressive. There are several factors which could have played a part in degrading the results. Poorly known robot kinematics and unmodeled/nonlinear noise are probably the largest contributors. Tolerances in the design of the manipulator's structure prohibit the precise calculation of the link lengths. These lengths can even change as the robot moves through its full range of motion. The LVDT which is used to measure the distance between the robot's wrist and the trackball also results in a very unprecise link length. Since this link is at the end of the chain it is magnified by all the other kinematic errors.

Noise can be introduced into the system through surface measurement errors. The surface normal errors are cross correlated due to the algorithms the force/torque controller uses to convert the strain gage readings into cartesian forces. The surface location errors are cross correlated because of the kinematic equations used to convert joint position into cartesian position. These measurement errors are further correlated when they are transformed from surface measurements into parameters. The Kalman filter's performance will be suboptimal unless its noise covariance matrix is properly constructed to reflect the measurement error's cross correlation. Thus, analyzing the system's measurement errors and how they are correlated might improve the shape estimates.

An alternative could be to reformulate the surface model equations in terms of the surface measurements, i.e. replace the nine quadric parameters with surface quantities that could be directly measured. The surface quantities do not necessarily have to be cartesian position and orientation. Instead, several possibilities could be the manipulator's six joint positions, the eight strain readings of the force/torque sensor, the deflection of the LVDT in the wrist, and the rotation rate of the trackball. An artificial neural network might even be useful in filtering this data. Given enough data, the network could try to distinguish the meaningful measurements from superfluous ones. It could then combine them into descriptive quantities.

Another possible contributing factor to the poor estimates could be the oscillating force control. The LVDT should detect the motion of the robotic arm and adjust the kinematic equations appropriately. However it undoubtingly introduces a small amount of position error. The force oscillations are likely to have a much larger effect on surface normal measurements. This is because the surface normal is derived from the sensed contact force. If the contact force is continuously changing, its directional vector is also likely to fluctuate.

The performance of the optimal trajectory generated on-line was inconclusive when compared against suboptimal and preprogrammed trajectories. In addition the poor performance of the estimator further obscured the results. The fact that some trajectories had to be stopped prematurely also hampered comparisons. These results could be improved if a method was found to prevent the manipulator from over flexing its wrist. This could be built into the optimal trajectory generator by formulating the chance of the wrist over flexing as a cost function. The trajectory generator would simply have an additional cost function to minimize.

The surface curvature measurements showed promise but the technique needs further refinement. The major problem areas were insuring good trackball rotation, mechanical tolerances, and filtering the data. Trackball rotation can be improved by adding a rough texture to the surface being explored, placing a rubber coating over the hard plastic trackball (however, this could introduce an unmeasurable deformation in the robot's tactile probe), and by smoothing out the contact force. On the filtering side, a good derivative filter would help edge detection.

Using this thesis as a basis, further research could be conducted to estimate multiple surface shapes such as standard or deformed spheres, cylinders, quadrics, superquadrics, and polyhedrons. This could be applied to modeling a single object or, using constructive solid geometry, a combination of several objects. In the case of a single object, the estimate with the most certainty would be selected at the end of exploration. For multiple objects, an artificial intelligence algorithm would be needed to transition the surface estimator from one object model to another. Reliable surface curvature measurements would be very useful in this instance. Discontinuities or sign changes in surface curvature can be used to locate boundaries between adjoining objects.

## REFERENCES


[1] Allen, P. K., "Mapping haptic exploratory procedures to multiple shape representations," *Proc. IEEE International Conference on Robotics and Automation*, pp. 1679-1684, Cincinnati, OH, May 1990.

[2] Allen, P. K. and K. S. Roberts, "Haptic object recognition using a multi-fingered dexterous hand," *Proc. IEEE International Conference on Robotics and Automation*, pp. 342-347, Scottsdale, AZ, 1989.

[3] Arman, F. and J. K. Aggarwal, "Object recognition in dense range images using a CAD system as a model base," *Proc. IEEE International Conference on Robotics and Automation*, pp. 1858-1863, Cincinnati, OH, May 1990.

[4] Barr, A. H., "Superquadrics and angle-preserving transformations," *IEEE Transactions on Computer Graphics and Animation*, pp. 11-23, January 1981.

[5] Bay, J. S., "An estimator based exploration engine for autonomous learning systems," *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1411-1416, Charlottesville, VA, October 1991.

[6] Bay, J. S., "A fully autonomous active sensor-based exploration concept for shape sensing robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 4, pp. 850-860, July/August 1991.

[7] Bay, J. S. and H. Hemami, "Dynamics of a learning controller for surface tracking robots on unknown surfaces," *IEEE Transactions on Automatic Control*, vol. 35, no. 9, pp. 1051-1054, September 1990.

[8] Bay, J. S., "Tactile shape sensing via single- and multi-fingered hands," *Proc. IEEE International Conference on Robotics and Automation*, pp. 290-295, Scottsdale, AZ, 1989.

[9] Berkemeier, M. D. and R. S. Fearing, "Determining the axis of a surface of revolution using tactile sensing," *Proc. IEEE International Conference on Robotics and Automation*, pp. 974-979, Cincinnati, OH, May 1990.

[10] Cameron, A., "Optimal tactile sensor placement," *Proc. IEEE International Conference on Robotics and Automation*, pp. 308-313, Scottsdale, AZ, May 1989.

[11] Delingette, H., M. Hebert, K. Ikeuchi, "Shape Representation and Image segmentation using deformable surfaces," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 467-472, Lahaina, Maui, HI, June 1991.

[12] Elosegui, P., R. W. Daniel, and P. M. Sharkey, "Joint servoing for robust manipulator force control," *Proceedings of the 1990 International Symposium on Intelligent Control*, 1990, pp. 246-251.

[13] Fan, T. J., G. Medioni, R. Nevatia, "Description of surfaces from range data using curvature properties," *Proc. IEEE Computer Society on Computer Vision and Pattern Recognition*, pp. 86-, Miami Beach, FL, June 1986.

[14] Fan, T. J., G. Medioni, R. Nevatia, "Segmented descriptions of 3-D surfaces," *IEEE Journal of Robotics and Automation*, vol. 3, no. 6, pp. 527-538, December 1987.

[15] Fearing, R. S. and T. O. Binford, "Using a cylindrical tactile sensor for determining curvature," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 806-817, December 1991.

[16] Fu, K. S., R. C. Gonzalez, C. S. G. Lee, Robotics: Control, Sensing, Vision, and Intelligence, New York: McGraw-Hill, 1987.

[17] Gaston, P. C. and T. Lozano-Pérez, "Tactile recognition and localization using object models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*," vol. 6, no. 3, pp. 257-, May 1984.

[18] Grimson W. E. L. and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *International Journal of Robotics Research*, vol. 3, no. 3, pp. 3-33, Fall 1984.

[19] Hogan, N., "Impedance control: an approach to manipulation, Parts I, II and III," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 1-24, March 1985.

[20] Hollinger, J. G. and J. S. Bay, "Autonomous shape exploration using simple sensors," *International Symposium on Robotics and Manufacturing*, Santa Fe, NM, November 1992.

[21] Hollinger, J. G., R. A. Bergstrom, and J. S. Bay, "A fuzzy logic force controller for a stepper motor robot," *Proc. IEEE International Symposium on Intelligent Control*, Glasgow, Scotland, August 1992.

[22] Ishikawa, H., C. Sawada, K. Kawase and M. Takata, "Stable compliance control and its implementation for a 6 D.O.F. manipulator," *Proc. IEEE International Conference on Robotics and Automation*, pp. 98-103, Scottsdale, AZ, May 1989.

[23] Kazerooni, H., B. J. Waibel, and S. Kim, "On the stability of robot compliant motion control: theory and experiments," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 417-426, September 1990.

[24] Kofakis, P. "Recognition and localization of objects using sparse data," *Proc IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp 647-650, Miami Beach, FL, June 1986.

[25] Kosko, B., Neural Networks and Fuzzy Systems, Englewood Cliffs: Prentice-Hall, 1992.

[26] Leahy, M. B., "Experimental analysis of robot control: a performance standard for the PUMA-560," *Proceedings of the IEEE Symposium on Intelligent Control*, Albany, NY, pp. 257-264, September 1989.

[27] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller - Part I," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-418, March/April 1990.

[28] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller - Part II," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419-435, March/April 1990.

[29] Lee, S. and H. Hahn, "Recognition and localization of 3-D natural quadric objects based on active sensing," *Proc. IEEE International Conference on Robotics and Automation*, pp. 156-161, Sacramento, CA, April 1991.

[30] Mills, J. K. and A. A. Goldenberg, "Force and position control of manipulators during constrained motion task," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 30-46, February 1989.

[31] Newman, W. M., and R. F. Sproull, Principles of Interactive Computer Graphics, Second ed., New York: McGraw-Hill, 1979.

[32] Nowlin, W. C., "Experimental results on Bayesian algorithms for interpreting compliant tactile sensing data," *Proc. IEEE International Conference on Robotics and Automation*, pp. 378-383, Sacramento, CA, April 1991.

[33] Okada, T. and U. Rembold, "Proximity sensor using spiral-shaped light-emitting mechanism," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 798-805, December 1991.

[34] Pentland, A. P. and M. Park, "Recognition by parts," *Proc. IEEE International Conference on Computer Vision*, pp. 612-620, 1987.

[35] Pentland, A. P., "Part models," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 242-249, Miami Beach, FL, June 1986.

[36] Raibert, M. H. and J. J. Craig, "Hybrid position/force control of manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 126-133, June 1981.

[37] Roberts, K. S., "Robot active touch exploration: constraints and strategies," *Proc. IEEE International Conference on Robotics and Automation*, pp. 980-985, Cincinnati, OH, May 1990.

[38] Sander, P. T. and S. W. Zucker, "Stable surface estimation," *Proc. IEEE International Conference on Pattern Recognition*, pp. 1165-1167, Paris France, October 1986.

[39] Sclaroff, S. and A. Pentland, "Closed-form solutions for physically-based shape modeling and recognition," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 238-243, Lahaina, Maui, HI, June 1991.

[40] Shimojo, M., M. Ishikawa and K. Kanaya, "A flexible high resolution tactile imager with video signal output," *Proc. IEEE International Conference on Robotics and Automation*, pp. 384-391, Sacramento, CA, April 1991.

[41] Sinha, S. S. and P. J. Besl, "Principle patches: a viewpoint-invariant surface description," *Proc. IEEE International Conference on Robotics and Automation*, pp. 226-231, Cincinnati, OH, May 1990.

[42] Solina, F. and R. Bajcsy, "Recovery of parametric models from range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 131-147, February 1990.

[43] Stockli, T., "Automatic robotic exploration for superquadric surfaces," Unpublished technical report submitted to faculty member J. S. Bay of the Virginia Tech Department of Electrical Engineering, August, 1991.

[44] Struik, D. J., Lectures on Classical Differential Geometry, New York: Dover Publications, Inc., 1961.

[45] Terzopoulos, D. and D. Metaxas, "Dynamic 3D models with local and global deformations," *Proc. IEEE International Conference on Computer Vision*, pp.606-615, Osaka, Japan, December 1990.

[46] Wen, J. T. and S. Murphy, "Stability analysis of position and force control problems for robot arms," *Proc. IEEE International Conference on Robotics and Automation*, pp. 252-257, Cincinnati, OH, May 1990.

[47] Wen, J. T., "Position and force control of robot arms," *Proceedings of the IEEE Symposium on Intelligent Control*, Albany, NY, pp. 251-256, September 1989.

[48] Whitney, D. E., "Historical perspective and state of the art in robot force control," *International Journal of Robotics Research*, vol. 6, no. 1, pp. 3-14, Spring 1987.

# APPENDIX

## PARAMETER VECTOR DECOMPOSITION

Given: a parameter vector, $P$, describing an ellipsoid.

Find: the length and translation of the ellipsoid three major axes.

Derivation:

$$S(X, P) = X^T E X + 2F^T X - 1 = 0$$

where $P = [a'\ b'\ c'\ d'\ e'\ f'\ u'\ v'\ w'\ ]^T$, $E = \begin{bmatrix} a' & b' & c' \\ b' & d' & e' \\ c' & e' & f' \end{bmatrix}$, $F = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix}$, *and* $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

Assume matrix $E$ has three independent eiganvectors.

Then, there exists a change of basis operator $\bar{Q}$ that will diagonalize $E$.

Let $\bar{X} = \bar{Q}X$.

$$S(\bar{X}, P) = S(\bar{Q}X, P)$$

$$S(\bar{X}, P) = (\bar{Q}X)^T E\bar{Q}X + 2F^T\bar{Q}X - 1$$

$$S(\bar{X}, P) = X^T\bar{Q}^T E\bar{Q}X + 2F^T\bar{Q}X - 1$$

$$S(\bar{X}, P) = X^T\bar{E}X + 2\bar{F}^T X - 1$$

where $\bar{E} = \bar{Q}^T E\bar{Q} = \begin{bmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & f \end{bmatrix}$ and $\bar{F} = F^T\bar{Q} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$.

$$S(\bar{X}, P) = ax^2 + dy^2 + fz^2 + 2(ux + vy + wz) - 1$$

Complete square:

$$S(\bar{X}, P) = a\left(x^2+2\frac{u}{a}x+\frac{u^2}{a^2}\right) + d\left(y^2+2\frac{v}{d}y+\frac{v^2}{d^2}\right) + f\left(z^2+2\frac{w}{f}z+\frac{w^2}{f^2}\right) - 1-\frac{u^2}{a}-\frac{v^2}{d}-\frac{w^2}{f}$$

$$S(\bar{X}, P) = a\left(x+\frac{u}{a}\right)^2 + d\left(y+\frac{v}{d}\right)^2 + f\left(z+\frac{w}{f}\right)^2 - \left(1+\frac{u^2}{a}+\frac{v^2}{d}+\frac{w^2}{f}\right)$$

$$S(\bar{X}, P) = a\left(x+\frac{u}{a}\right)^2 + d\left(y+\frac{v}{d}\right)^2 + f\left(z+\frac{w}{f}\right)^2 - q$$

where $q = 1+\frac{u^2}{a}+\frac{v^2}{d}+\frac{w^2}{f}$.

Place in standard form of an ellipsoid:

$$a\left(x+\frac{u}{a}\right)^2 + d\left(y+\frac{v}{d}\right)^2 + f\left(z+\frac{w}{f}\right)^2 = q$$

$$\frac{\left(x+\frac{u}{a}\right)^2}{\frac{q}{a}} + \frac{\left(y+\frac{v}{d}\right)^2}{\frac{q}{d}} + \frac{\left(z+\frac{w}{f}\right)^2}{\frac{q}{f}} = 1$$

$$\frac{(x-\bar{x}_0)^2}{(r_{\bar{x}})^2} + \frac{(y-\bar{y}_0)^2}{(r_{\bar{y}})^2} + \frac{(z-\bar{z}_0)^2}{(r_{\bar{z}})^2} = 1$$

where $\bar{x}_0 = -\frac{u}{a}$, $\bar{y}_0 = -\frac{v}{d}$, $\bar{z}_0 = -\frac{w}{f}$, $r_{\bar{x}} = \sqrt{\frac{q}{a}}$, $r_{\bar{y}} = \sqrt{\frac{q}{d}}$, $r_{\bar{z}} = \sqrt{\frac{q}{f}}$.

$$X_0 = \bar{Q}^{-1}\bar{X}_0$$

## VITA

James Hollinger was born February 20, 1968 in Washington D. C. After he received his Bachelor of Science in Electrical Engineering from Virginia Tech in May of 1991, he remained at Virginia Tech to pursue a Masters degree in the same discipline. Three CO-OP sessions with Martin Marietta Corporation in Baltimore Maryland helped develop an acute interest in automatic control systems, robotics, and neural networks. While studying these subjects for a Masters degree, he setup a MERLIN 6540 Industrial Robot and interfaced it to an IBM compatible PC. This robot was then used to perform Thesis research. Mr. Hollinger is now a member of the Pegasus® program at Orbital Sciences Corporation in Dulles Virginia.

James G. Hollinger