Grasp Gaits for Planar Object Manipulation

by
Susanna Richmond Leveroni

M.S., Boston University (1991)
B.A., Yale University (1986)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1997

Signature of Author . . . .
Department of Mechanical Engineering
September 13, 1996

Certified by . . . . [illegible] . . . .
Dr. Kenneth Salisbury
Principal Research Scientist

Accepted by . . . .
Prof. Ain Sonin
Chairman, Departmental Committee on Graduate Students

Grasp Gaits
for
Planar Object Manipulation

by

Susanna Leveroni

Submitted to the Department of Mechanical Engineering
on September 13, 1996, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Pick up a pen, and with your eyes closed, twirl it around in your fingers. How do you know where to put your fingers? How do you know what sequence of grasps and intermediate object motions will bring the pen to the final desired location without dropping it? What kind of sensory information are you using and how are you interpreting it? Would you use the same approach if you tried to do this with an apple?

Robotic researchers have built a number of dexterous robot hands designed to grasp and manipulate a variety of objects. While these hands have been able to mimic the finger motions of human hands, the problem of enabling such devices to autonomously manipulate objects through significant displacements still remains largely unsolved. There is a conspicuous lack of understanding of how to plan such motions and how to monitor and correct the execution of the plans.

This thesis addresses the specific question of how to reorient convex objects in the plane using a three-fingered robot hand. Since the workspace of any robot finger is limited, a large-scale object reorientation will require a series of smaller object motions and regrasps, known as a *grasp gait*. This thesis deals with the planning of such grasp gaits for large-scale object reorientation and focuses on three important questions. The first question deals with how to generate a gait for continual rotation given a specific convex object and set of finger workspaces. For cases where the finger workspaces are small, the coefficient of friction is low, or for certain types of objects which are difficult to manipulate, many grasps and object orientations lead to dead ends, where no new grasp can be found such that rotation can continue. Answering the first question provides a way to navigate through these choices. The second question focuses on how these gaits can be understood within some general framework. Understanding why a gait does or does not exist for a particular object, or being able to classify gaits into general types provides information which can be used to predict what type of solution will apply to a specific problem or to modify the problem if no solution exists within its present constraints. Finally, this thesis deals with the issue of implementation. Using a few simple control algorithms and only position sensor information we demonstrate that these gaits are robust and can be implemented on an actual robot hand without having to replan.

Thesis Supervisor: Dr. Kenneth Salisbury
Title: Principal Research Scientist

Acknowledgments

I would like to thank my advisor, Dr. J. Kenneth Salisbury, for allowing me the freedom to pursue my ideas for this thesis and for dreaming with me about its possibilities. I would also like to thank him for his technical guidance and for his attitude that hard work is one - not the only - ingredient for a successful life. I would like to thank Professor Tomas Lozano-Perez for his weekly meetings with me. His ability to turn a confusing question into a well-posed problem always inspired me, and his guidance renewed my enthusiasm and provided the structure for a major portion of this work. I would like to thank Professor Kamal Youcef-Toumi for his support, teaching and practical insights both as a part of the Mechanical Engineering department and as a member of my committee.

Several members of the lab deserve a special thanks. Thanks to John Morrell for his warm welcome to the lab and for always being a positive voice. Thanks to Tom Stahovitch for his time in helping me prepare for my committee meetings and for sharing his healthy attitude with me. Thanks to Brian Anthony for bringing the lab together socially, making it a fun place to be, and for cheering me up when I was discouraged. Thanks to Ron Wilkins for helping me whenever I had to build anything! Thanks to Chris Tarr and Chris Foley for all the Phantom support, and to Tom Massie for its development. Thanks to Dan Hagerty for all of his computer support. Thanks to Jacqui Taylor for help with so many of the details. Thanks also go to Vinay Shaw, Craig Latimer, Donald Green, Brian Eberman and Dave Brock for everything from helpful technical discussions to Friday night happy hours. More recently, thanks to Lisa Kozsdiy and all of my '7th floor friends' for making the lab a much happier, healthier place!

Several people outside of the lab also deserve mention. I would like to thank Ling Chang for her loyal friendship and for her many encouraging calls to the lab at all hours of the night when I was trying to finish up. I would like to thank Hugh Herr for helping me formulate my ideas, for studying with me, and for always being there when I needed him. I would also like to thank Kevin Brown for his immense technical and emotional support throughout my stay at MIT and for having such enthusiasm for my work.

I would also like to thank my family for their love and support throughout and leading up to this process. I would like to thank my mom for always believing that I could do anything that I set my mind to. I would like to thank my sister, Jane, for helping me keep my perspective and my brother, Ted, for telling me when it was time to quit!

Finally, I would like to thank Mattan Kamon for his endless patience with me during those final months, for staying up with me while I was preparing my defense, for encouraging me, and most of all, for giving me so much to look forward to.

Contents

Chapter 1

Introduction

1.1 The Development of Dexterity

Human beings are dexterous creatures. Most of what we see around us - our offices, our homes, our cities - exists because we have the ability to manipulate things with our hands. Yet we do not understand exactly what it is we do when we manipulate an object within our grasp. As our understanding of manipulation has become more abstract, we have been able to build tools which in turn increase our own dexterity. As we discover the fundamental principles that govern the task of manipulation, we will be able to build robots that are dexterous themselves.

At three months a child can learn to grasp a bottle. A dog can learn to open a door with his paw. While both the child and the dog may have made some connection between a gross motor behavior and a desired result, presumably neither is aware of the planning, sensing, and control which they have learned to execute in order to accomplish the task.

Our early ancestors fashioned simple carving tools as far back one million years ago. Present day chimpanzees have been observed stripping leaves from a twig in order to use it to pull edible insects from a hole. While simple toolmaking requires a higher level of abstraction than does simple manipulation (requiring the knowledge or imagination that if we had something harder, sharper or more pointed which does not yet exist, we could move it in such a way as to achieve a desired goal), the tool user is still required to perform all of the planning, sensing and control necessary to use the tool and is at least partially unaware of how he does this.

As we have learned what is required to position and move things precisely, we have been able to build machines which move more accurately or forcefully than we ourselves do. Using such machines as the lathe and milling machine, we have been able to create parts accurately enough that they became interchangeable, allowing for the development of the assembly line and leading to the Industrial Revolution. While some low-level motion control can be performed by machines and computers, a human is still often required to set up initial position of the manipulated object and to intervene or even detect if anything unexpected occurs.

Now we are trying to build dexterous robots. We would like to build robots that can grasp and manipulate a variety of objects without having to change the hardware involved or the programs that run them. This ability requires the knowledge to generalize between tasks and deal with some amount of uncertainty. In order to give the robot this knowledge, we must be able to represent the task of manipulation in some general form, understand what kind of reasoning or decision making might be required during the process, know what actions are required to carry out those decisions, and be able to implement those actions. As we are successful in doing this, robots will begin to perform manipulation tasks that humans can and perhaps some that we cannot.

1.2 A Specific Type of Manipulation

Let us consider how we perform a specific type of manipulation - the task of reorienting an object within our grasp when the amount we want to move the object is greater than our fingers can move. For instance, if we pick up an object such as a pen, but it is oriented incorrectly, how are we able to move it to the correct position? Most likely we would move the pen to an intermediate position and then switch to a new grasp so that we could move the pen further. This sequence of finger motions and regrasps we call a *grasp gait*. While we may not be able to describe exactly how we accomplish this task, we can identify several subtasks which we are able to perform, consciously or unconsciously, in order to achieve our goal.

1) We know how to move our fingers to cause the object to move in some general desired direction.
2) We know how to interpret sensory information, from touch and possibly vision, to estimate where the object has moved and to plan for the next motion.
3) We know how to switch to a new grasp when necessary which will allow us to continue the motion.
4) We know roughly what combination of local motions and regrasps will bring the object to the final desired location.
5) While performing all of these actions we are able to maintain a stable or quasi-stable grasp on the object so that we do not drop it.

1.3 General Thesis Questions

This thesis addresses a similar list of questions in order to use a robot hand to reorient objects using grasp gaits. Specifically, the problem of reorienting convex objects in a plane is considered, where all finger contacts and forces acting on the object are constrained to lie in that plane as well. (See figure 1.1.) If friction is considered, it requires only 2 fingers to constrain an object in a plane, but to switch from one stable grasp to another requires a third finger. Therefore this thesis considers grasp gaits for a three-fingered robot hand, the minimum number to accomplish this task.

The first set of questions addresses the problem of grasp gait planning.
1) How can a sequence of local motions and regrasps be found which will result in the final desired object orientation?

2) Is there a guarantee that a new stable grasp will always exist given a specific object geometry, finger workspace and grasp such that motion can continue (See figure 1.2.)?

3) If not, could something have been done in previous steps to ensure that a new grasp would exist?

4) Is there a guarantee that some gait will exist to reorient a given object a desired amount?

5) Is there a way to classify different types of objects and the gaits which reorient them and would such a classification simplify the task of finding a sufficient gait?

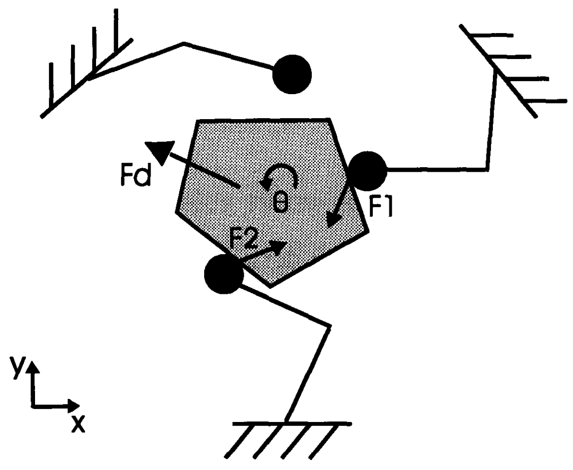

Figure 1.1 An illustration of a three-fingered robot hand reorienting a planar object.

The second set of questions addresses the problem of implementing the planned gait using an actual robot hand.

1) What kind of control algorithms and sensors are necessary to achieve the correct finger motions and forces to cause the object to move in the desired way?

2) Using only finger position sensors can object position and grasp errors be detected?

3) How can actual object and grasp positions relative to the object be estimated?

4) How can the current gait be modified to correct for these errors?

In addressing these questions a grasp gait planner is developed for manipulating a large class of objects in the plane and the results are successfully implemented using a three-fingered robot hand.

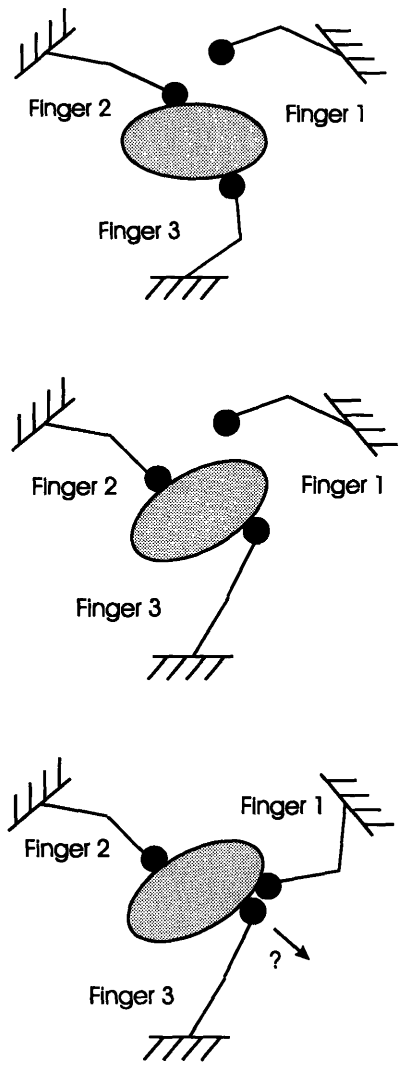

Figure 1.2 A series of motions demonstrating that a new grasp is not always possible. The first image shows a grasp between fingers 2 and 3. The second shows the same grasp after the object has been rotated until finger 3 reached the end of its workspace. The third image shows that a new grasp between finger 1 and 2 may not be found such that finger 3 can be removed and motion can continue. If the coefficient of friction is not large enough, this grasp between 1 and 2 will not be stable. Is there a position where finger 1 can be placed such that finger 2 can be removed and then placed in a new location such that finger 3 can be finally removed? There is in fact a grasp between 1 and 2 that exists and is stable (such as the near the two endpoints of the major axis), although there may be no way of reaching that grasp through a succession of stable grasps beginning with the one shown.

Chapter 2

Background

2.1 Overview

This chapter reviews some of the relevant work done in the field of grasping and manipulation. Robotic researchers have long sought to build and utilize articulated end-effectors, or hands, to extend robotic grasping and manipulative capability. Simple one-degree-of-freedom grippers are limited to grasping a small range of objects and cannot impart any motion to the object. By employing multiple, articulated fingers in the end-effector design it becomes possible to stably grasp a wider range of objects. Furthermore, if sufficient freedom in the hand/object system exists, then the fingers may be used to impart controlled motions to the grasped object. The potential increase that such designs promise has motivated an enormous amount of research into how to build and use robot hands. Section 2.1 summarizes some of this work. Section 2.3 presents a more detailed summary of the definitions and techniques that are particularly relevant to this thesis. Section 2.4 outlines the remaining chapters of this thesis which are the original contributions to this field.

2.2 Related Work

This section summarizes some of the previous work done in the areas of building robot hands, finding stable grasps of an object, optimizing grasps, and manipulating objects using a robot hand.

Hardware:

Many robotic hands have been designed and built. Among the better known articulated hands is the Salisbury Hand, which has three-fingers, each with three degrees of freedom, designed to optimize the fingers' ability to apply forces [20]. Another is the anthropomorphic, five-fingered Utah/MIT hand, with links and joints similar to humans [9]. Okada developed designs for hands with redundant degrees of freedom in each finger [16].

Stable Grasps:

Much work has been done in the area of defining stable grasps. Reuleaux showed that 4 frictionless contacts were necessary to constrain an object in the plane (known as form closure). Somov showed that 7 contacts were necessary to constrain an object in 3-d space. Lakshminarayana presented a method for determining if a specific set of 7 contacts are sufficient for form closure [11]. Salisbury presented a method for determining if an arbitrary set of contacts, which could include friction, were sufficient for constraining an object using the grip matrix [13]. Nyugen developed a geometric test to determine if 2 point contacts with friction could resist arbitrary disturbance forces in the plane, a condition known as force closure [15]. Hanafusa and Asada considered whether grasps were stable in a dynamic sense, by evaluating the stiffness matrix of the grasp controller [7].

Optimal Grasps:

Much work has also been done in the area of finding optimal grasps. Kerr and Roth presented a method to choose the set of internal grasp forces for a given grasp such that the total finger forces were furthest from violating friction cone and joint torque limit constraints [10]. Ferrari and Canny developed an algorithm for choosing a grasp which either minimizes the maximum finger force or the total finger force required to resist a disturbance force [5]. Li and Sastry considered these quality measures as well as one which considers how well suited a grasp is to a particular task [12]. Pollard presented a method for extending grasps suited for a particular object and task to objects of similar type [18].

Manipulation:
There are several different types of manipulation that have been studied in the literature. Salisbury investigated how small object motions can be made given a particular grasp, using the remaining degrees of freedom of the hand [13]. Brock considered a particular grasp and presented a method for causing the object to slip in a controlled way [2]. Abell and Erdmann [1] as well as Rus [19] considered reorienting objects with frictionless contacts using controlled slip. Goldberg used only a parallel jaw gripper to reorient planar polygons using slip [6]. Kerr and Roth considered the kinematics of rolling contacts [10]. Montana presented a general form in which to consider the full kinematics of the finger-plus-object system [14].

Several authors have also begun to look at using grasp gaits of stable grasps to reorient objects. Fearing looked at a specific gait to twirl a baton and implemented it on a Salisbury Hand [4]. Hong et al considered using one grasp to reorient an object by replacing one finger with another [8]. Omata and Nagata presented a planner which reorganized the fingers such that a finite object reorientation would be possible [17]. Chen and Burdick developed a method for solving and representing all stable grasps of a smooth object, suggested that the representation could be used for finding grasp gaits, and demonstrated a few examples [3].

2. 3 Definitions and Techniques

This section reviews some of the terminology and methods that are well known in the literature and important to this thesis. The topics that are covered here include contact types, grasp types, friction cone limits, complete restraint, and the grasp map.

Stable Grasp:

In order for an object to be manipulated, it must be stably grasped. A stable grasp is a set of finger contacts through which a set of forces and moments are applied such that the object is held in static equilibrium or constrained to move in a desired way. We call a set of forces and moments applied to the object by a contact a *contact wrench,* and all other forces and moments a *disturbance wrench.*

Contact Types:

We differentiate between contact types according to the shape of the contact area and the effect of friction at the contact. A *point contact* is one whose contact area is effectively zero. If a point contact is frictionless, it can apply a force through the contact point and only in the direction of the object's inward normal at that point. If it is a point contact with friction, it can also apply a force in any direction tangential to the surface. A point contact can pivot about the point of contact on the object, so the point of contact relative to the object remains fixed during object motion. Figure 2.1 shows an example grasp using point contacts and illustrates the kinematics during motion.

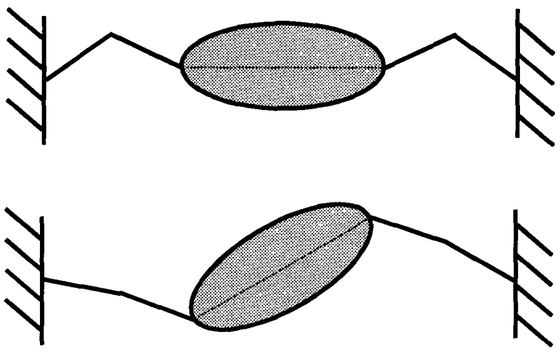

Figure 2.1 A two-fingered grasp of an ellipse using point contacts. As the object is rotated, the contact points relative to the object remain unchanged.

A *hard finger contact* also has a contact area that is effectively zero, and therefore can apply the same forces as a point contact. It differs from a point contact in that it rolls along the surface of the object during object motion, so the contact point on both the object and the finger moves during motion. Figure 2.2 shows a grasp using hard finger contacts as well as the resulting kinematics during motion.

A *soft finger contact* is considered to have a contact area that is nonzero, so in addition to applying the forces that a point and hard finger contact can, it can also apply a moment normal to the contact surface if friction is present. The contact area is considered to be small enough, however, that it allows rolling at the contact by the finger. If, however, the radius of the finger is significantly less than the radius of the object at the point of contact, it may be modeled as a point contact with regard to motion. Figure 2.3 shows the same grasp using soft finger contacts with the same kinematics, whereas figure 2.4 shows the grasp with much smaller fingertips, where rolling is negligible. The above definitions are given in [Salisbury] . This thesis predominantly considers point contacts but considers hard-finger contacts where rolling is negligible as well.

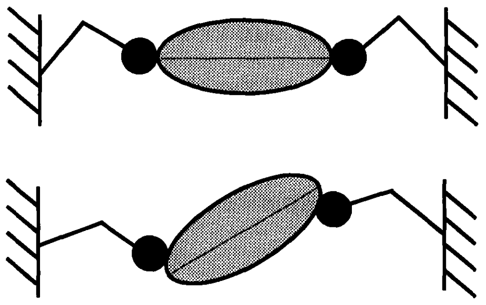

Figure 2.2 A two-fingered grasp of an ellipse using hard finger contacts. The lower figure shows that the contact points move as the object is rotated.

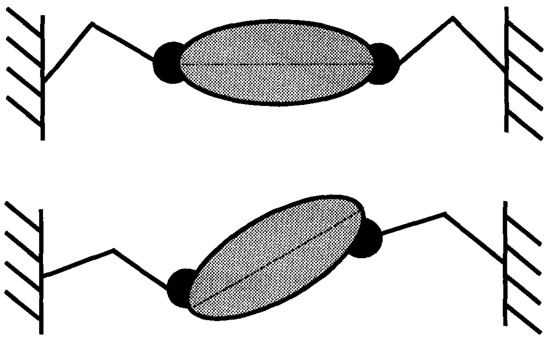

Figure 2.3 A two-fingered grasp using soft contacts, where the fingertips conform to the object surface. The same rolling occurs as with hard contacts.

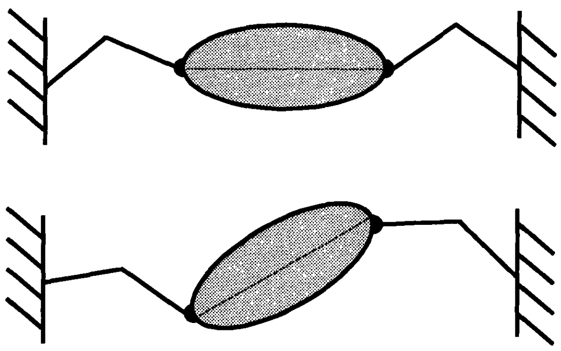
Figure 2.4 A two-fingered grasp using soft contacts, where rolling is negligible due to the small ratio between fingertip and object radius.

Friction Cone Limits:

Both tangential forces and moments normal to the surface can only be exerted if friction is present. Considering only tangential forces, the tangential force exerted through a contact is constrained to be less than or equal to the product of the normal force and the coefficient of friction between the two surfaces, μ. Geometrically, the resultant force must lie within the contact's friction cone, which is defined by the boundary where the tangential force is equal in magnitude to the product of the normal force and the coefficient of friction. (See figure 2.5.) The friction cone can be defined by an angle θ, called the friction angle, where

$$\theta = \tan^{-1}(ft_{\max} / fn) = \tan^{-1}(\mu\, fn / fn) = \tan^{-1}(\mu).$$

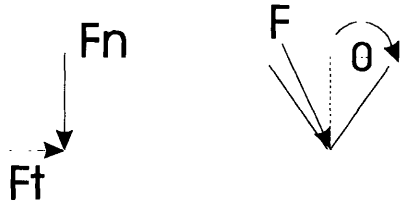

Figure 2.5 A resultant force F, shown broken into its tangential and normal components and also lying within its friction cone.

In the 3-d case, moments normal to the contact may be resisted in the case of soft-finger contacts. In this thesis we will be primarily considering the planar case.

Types of Stable Grasps:

An example of a stable grasp using frictionless point contacts is shown in Figure 2.6. The sum of contact wrenches is equal and opposite to the gravity wrench. Clearly if the disturbance wrench was in a direction other than the gravity wrench, this set of contacts would be insufficient to constrain the object, because the fingers can only push, and not pull, on the object.

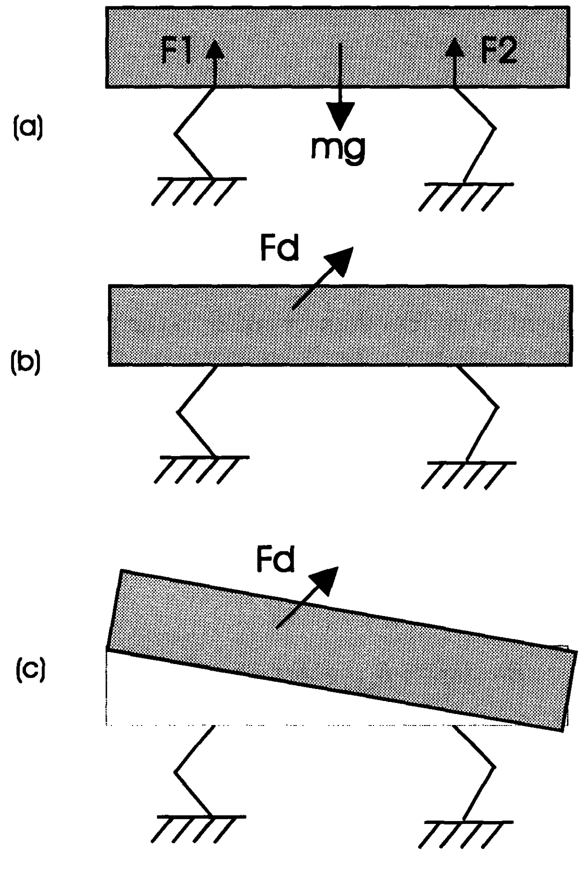

Figure 2.6 A series of figures showing various grasp types. Figure 2.6 (a) shows a grasp that is stable when the only disturbance force acting on the object is that of gravity. Figure 2.6 (b) and (c) show that when the disturbance force is changed, the grasp becomes unstable because the fingers cannot pull on the object.

A grasp is defined to be *force-closure* if its set of contacts can apply an arbitrary wrench to the object, thus having the ability to resist any disturbance wrench applied to it. A grasp is considered to be *form-closure* if the object can be fully constrained by the set of contacts, regardless of the force magnitudes applied. Since frictional forces depend on normal force magnitudes, a grasp which relies on friction may be force-closure but not form-closure. All form-closure grasps are force-closure, but the reverse is not true. As form-closure grasps rely purely on the structure of the contacts to constrain the object, they are sometimes called structural restraint grasps. (See pictures 2.7 and 2.8.) In this thesis we will consider force-closure grasps.

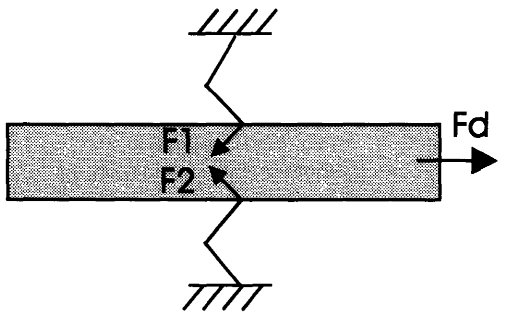

Figure 2.7 An example of a force closure grasp using 2 point contacts with friction. The fingers rely on friction to apply forces *F1* and *F2* which resist the disturbance force *Fd*.

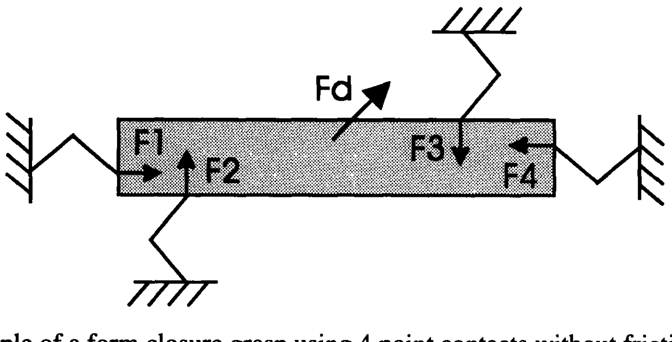

Figure 2.8 An example of a form closure grasp using 4 point contacts without friction. The disturbance force is resisted from the geometry of the contact positions.

Complete Restraint:

In general, a given grasp can be determined to be stable in the following way [Salisbury]. Each contact has a set of independent wrenches that it can apply to the object (e.g. a normal force, a tangential force, a normal moment). Figure 2.9 illustrates an example grasp showing the types of independent wrenches each contact can apply.

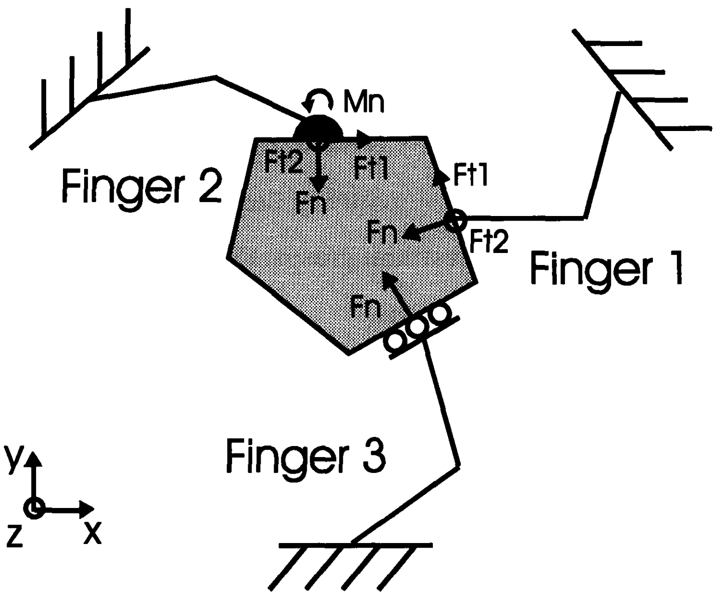

Figure 2.9 An 3 fingered grasp showing the different kinds of contact wrenches resulting from different contact types. Finger 1 is a point contact with friction and can apply 3 independent wrenches: a normal force and two tangential forces. Finger 2 is a soft contact with friction and can apply 4 wrenches: a normal force, 2 tangential forces and a normal moment. Finger 3 is a frictionless contact and can only apply a normal force.

A specific wrench can be represented in a column vector wi, where each element in the vector is the force or moment component of the wrench represented in a common reference frame. The set of wrenches which a given grasp can apply to the object is the union of the sets of wrenches for each contact. This new set can be represented in a matrix *W*, where each column of *W* is one of the wrenches that can be applied by one of the fingers.

The number of rows of the matrix *W* is equal to the number of degrees of freedom of the object, or alternatively, the number of independent basis wrenches that can be applied to the object which would cause the object to move. In general, for the planar case, the matrix *W* has 3 rows, since an x and y force and a z moment can be applied to cause motion. For the three dimensional case, *W* generally has 6 rows, since there are 3 independent forces and 3 moments that can be applied to cause motion. For example, the normal wrench of finger 3 would be represented as

$$\underline{w}_i = [Fx\ Fy\ 0\ 0\ 0\ Mz]^T,$$

since the force applies no force in the z direction and no moments about the x or y axes. If there are n contact wrenches, then

$$W = \left[\underline{w}_1\ \underline{w}_2\ \underline{w}_3 \cdots \underline{w}_n\right],$$

where $\underline{w}i$ is one of the independent contact wrenches that can be applied by one of the fingers. If the intensity of each wrench, *ci*, is written in a column vector $\underline{c}$, then a disturbance wrench $\underline{w}$ can be resisted when some $\underline{c}$ exists such that

$$W\underline{c} = \underline{w}.$$

Specifically, $\underline{c}$ can be written as

$$\underline{c} = \underline{c}_p + \lambda_1 \underline{c}_{h1} + \lambda_2 \underline{c}_{h2} + \cdots + \lambda_q \underline{c}_{hq},$$

where $\underline{c}p$ is a particular solution to equation 3.1, $\underline{c}hi$ is the *ith* independent homogeneous solution, λi is the magnitude of the *ith* homogeneous solution, and q is the number of homogeneous solutions and equal to the dimension of the null space of *W*. The homogeneous solutions are referred to as internal forces, since they act on the object without disturbing its equilibrium. An intuitive example of an internal force is the equal and opposite force that a pair of fingers can exert in the direction of the other. (See figure 2.10.). No matter how tightly these fingers squeeze, the net wrench applied by the fingers remains unchanged.

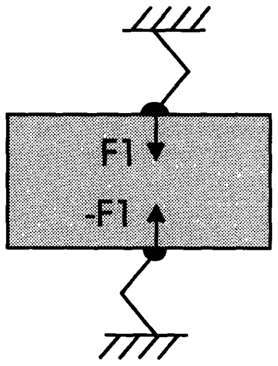

Figure 2.10 An example of two fingers applying an internal force to an object. No matter how tightly the fingers squeeze, the object's equilibrium is not disturbed.

If there were no constraints on the values of $\underline{c}i$, then a solution would always exist when the rank of *W* equaled the dimension of $\underline{w}$. A solution would also exist if the rank of *W* were less than the dimension of $\underline{w}$, provided that $\underline{w}$ lies in the column space of *W*.

However, there are constraints on the values of $\underline{c}i$. Normal forces can only be applied in the inward normal direction, and thus the intensities of these forces must be positive. (A contact can only push, rather than pull, on an object, unless the object is sticky.) In addition, tangential force intensities must be less than or equal in magnitude to the product of normal force intensities and the coefficient of friction as defined earlier. Similarly, normal moment intensities must be less than or equal in magnitude to the product of normal force intensities and a normal moment coefficient of friction. A solution $\underline{c}$ to equation 2.3 does not automatically satisfy the above constraints. In general, a system of linear inequalities must be solved to determine if a set of λi's exist such that the above constraints are met, implying that a given grasp is stable. (A specific example will be worked out in chapter 3.) In this thesis force-closure grasps are sought which also allow the object to be regrasped as it is reoriented. In many cases optimal grasps as defined in previous research are not sufficient in that they do not allow for reorientation.

The Grasp Map:

The set of stable grasps for a given object can be considered using the grasp map. It is similar to the contact configuration space suggested by [3] and will be briefly introduced here. For a given planar convex object, any point along its contour can be specified by one parameter, the angle that its position vector from a reference frame *f* makes with the horizontal axis. Any two-fingered grasp for that object can then be represented by a pair of angles and represented in a two-dimensional space, where the horizontal coordinate gives the position on the object of a finger *i*, and the vertical gives the position of a finger *j*. (See figure 2.11.) A grasp map can be constructed by numerically solving (point by point) for the regions in the map which represent force-closure grasps. In figure 2.11 the regions representing force-closure grasps are shaded.

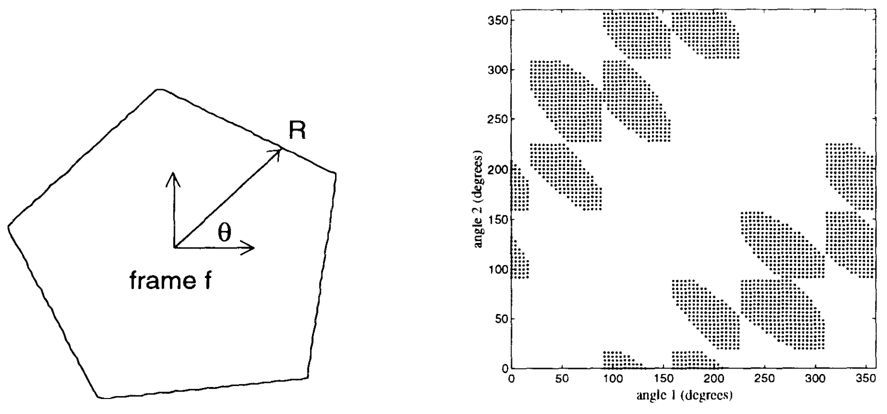

Figure 2.11 A convex object and its associated grasp map.

2.4 Thesis Outline

While researchers have begun to look at grasp gaits for reorienting objects, most have looked only at specific examples of gaits, or their methods are both unnecessarily complicated and not robust. There is still no general approach to generating and implementing gaits which can guarantee that if a solution exists it will be found, or it not, what would have to change in order for a gait to exist. Further, there is no approach which classifies the type of gait that is generated within some general framework. In addition, much of the work done in planning grasps and motions has been very theoretical and difficult to implement. This thesis seeks to develop a planner which is robust yet simple enough that it can be relied upon to generate the appropriate actions given any desired object trajectory, implement the desired trajectories and modify the plan when errors occur during implementation.

As a means to this end, this thesis analyzes the *planar* case of manipulating 2-dimensional star-shaped objects using point contacts with a three-fingered robot hand. An object is star-shaped if, for a given reference frame, any ray emanating from its origin crosses the object's contour only once. Therefore convex objects are a subset of star-shaped objects. A gait planner is developed which generates the sequence of finger motions and regrasps necessary to achieve the desired reorientation for a given object. Control algorithms necessary to implement these results on an actual robot hand are developed as well, achieving closed-loop control at the task-level using minimal sensory information.

In chapter 3 grasp maps for a variety of convex objects are generated, showing the stable regions for two-fingered grasps. An analysis of how varying global object properties such as eccentricity and smoothness effects the shapes of the stable regions in grasp map space is given. Also considered is how varying the coefficient of friction effects these shapes. In chapter 4 finger workspace limits are discussed and workspace maps are constructed, similar to grasp maps. It is shown how these two maps can be used together as a tool to represent and search or solve for grasp gaits to reorient objects. In chapter 5 a variety of search techniques are investigated to find grasp gaits which achieve a desired reorientation. Their usefulness and their limitations are discussed. In an effort to understand and classify the different possible gait patterns, in chapter 6 the global properties of the stable grasp regions in grasp map space are reconsidered, as they relate to global object properties. These features are used to develop prototype gaits which can be used to reorient similar classes of objects. It is shown that by finding simple patterns within the grasp map of a large class of objects, rule-based gaits can be developed based on these patterns and therefore extensive searches are not necessary to find a solution. Necessary conditions for these prototype gaits to exist are developed. These rule-based solutions are then used to prune the search tree for those cases where the sufficient conditions for the rule-based solutions are not met. In chapter 7 the implementation of these grasp gaits using an actual robot hand is discussed. The overall control structure of the system given, along with a detailed description of its components. An object localization algorithm is developed using position sensory information, so that when a gait is implemented errors can be detected and corrected. An active grip force adjuster is included, which keeps the finger forces within their friction cones while resisting external disturbance forces. This force adjuster relies only position sensor information as well. In chapter 8 the contributions of this thesis are summarized and its extensions to future work are discussed.

Chapter 3
Grasp Maps

3.1 Overview

In this chapter the *grasp map*, described briefly in chapter 2, is used to graphically represent all possible stable grasp regions for a given object. From this tool a qualitative understanding is gained of how stable grasp regions vary according to object shape. In addition, it also allows us to look for grasp gaits in a tractable way as well as to understand why they succeed or fail. In this chapter grasp maps are developed for 2-fingered point contact grasps of two-dimensional convex objects, where all forces are constrained to lie in the plane. In section 3.2 a method for generating grasp maps is presented. In section 3.3 the relationship between object eccentricity and the global shape of the stable regions of the grasp map is explored. In section 3.4 the relationship between the smoothness of an object and global grasp map shape is explored. In section 3.5 a number of interesting features of the grasp map is presented.

3.2 Developing the Grasp Map

We begin by developing the grasp map for two dimensional convex objects, using an approach similar to the *contact configuration space* developed by [Burdick]. As described in chapter 2, any point along the contour of a convex object can be specified by the angle that its position vector from a reference frame *f* makes with the positive horizontal axis. If the reference frame *f* is specified to lie within the object's contour and is fixed with respect to the object, there is a one-to-one mapping between an angle and a point on the contour. In fact, this is true for a broader class of objects, known as star-shaped objects, as discussed in chapter 2. The techniques developed in this thesis to generate grasp gaits are applicable to all star-shaped objects. Any two-fingered grasp for such an object can then be represented with respect to the object by a pair or angles. All possible two-fingered grasps for that object can be represented in a two-dimensional space, where the horizontal coordinate gives the position on the object of a finger *i*, and the vertical coordinate gives the position on the object of a finger *j*. (See figure 2.11.)

To determine whether a particular two-fingered grasp is force-closure, we use the approach presented in chapter 2. We seek a solution to the equation

$$W\underline{c} = \underline{w},$$

where W is the matrix for the given grasp made up of its contact wrenches, $\underline{c}$ is the vector of contact wrench intensities, and $\underline{w}$ is any disturbance wrench in 2-d space. A point contact with friction in the plane can apply a force both normal and tangential to the surface. Therefore a two-fingered grasp of this contact type has 4 independent wrenches that it can apply. Each vector has 3 components since there are 2 forces (x and y) and 1 moment (z) that can be applied by any wrench in 2-d. Equation 3.1 becomes

$$\left[\underline{w}_{1n}\ \underline{w}_{2n}\ \underline{w}_{1t}\ \underline{w}_{2t}\right]\underline{c} = \underline{w},$$

where the first 2 columns of W are the normal grasp wrenches and the last two are the tangential wrenches. If G is of rank 3, then there is one independent homogeneous solution and $\underline{c}$ can be written as $\underline{cp} + \lambda\, \underline{ch}$. The constraints on $\underline{c}$ are that

$$\begin{aligned} &c1 \geq 0, \\ &c2 \geq 0, \\ &c3 \leq \mu c1, \\ &c4 \leq \mu c2, \end{aligned}$$

since the normal forces cannot be negative and the tangential forces must be within their friction limits. Substituting in the above expression for $\underline{c}$, the constraints become

$$\begin{aligned} &cp1 + \lambda\, ch1 \geq 0, \\ &cp2 + \lambda\, ch2 \geq 0, \\ &\lambda(\mu\, ch1 - ch2) \geq cp2 - \mu\, cp1, \\ &\lambda(\mu\, ch3 - ch4) \geq cp4 - \mu\, cp3. \end{aligned}$$

If we seek a force-closure grasp, then W must be of rank 3, assuring the existence of a particular solution. In addition, the constraints become

$$\begin{aligned}&\lambda\, ch1 \geq 0,\\ &\lambda\, ch2 \geq 0,\\ &\lambda(\mu\, ch1 - ch2) \geq 0,\\ &\lambda(\mu\, ch3 - ch4) \geq 0,\end{aligned}$$

ensuring that regardless of the value of cp, a large enough value of λ can be chosen such that the previous constraints are met.

If an object contour is discretized into a finite number of points, a grasp map can be generated by testing each pairing of points for force-closure. For the grasp maps that follow, the object contours are discretized into 180 points and the coefficient of friction is specified in each example. The shaded regions represent force-closure grasps.

3.3 Object Eccentricity and the Grasp Map

In the following sections we develop a qualitative understanding of how an object's shape effects the stable regions of it's grasp map. In general, grasp maps are a function of the shape of the object and the coefficient of friction between the object and the fingers. With regard to shape, there are two parameters that we have used to characterize an object, where varying these parameters leads to distinctive types of grasp maps. The first parameter we use is the eccentricity of the object. For our purposes, we define the eccentricity, *E*, to be equal to the ratio of the largest object radius to the smallest radius, where the radius is measured from a point on the object's contour to the center of the object reference frame *f*. We begin with a circle, an eccentricity of 1, and a coefficient of friction $\mu = .3$. The object and corresponding grasp map is shown in figure 3.1. The grasp shown on the object is shown on the grasp map as an x. We see that if both fingers were to increase their angles by the same amount, shown on the grasp map as moving along a line with a slope of positive 1, the grasp would remain stable. (See figure 3.2.)

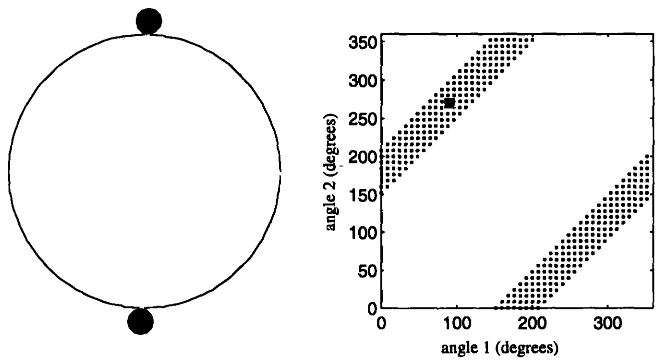

Figure 3.1 A two-fingered grasp of a circle and its corresponding grasp map.

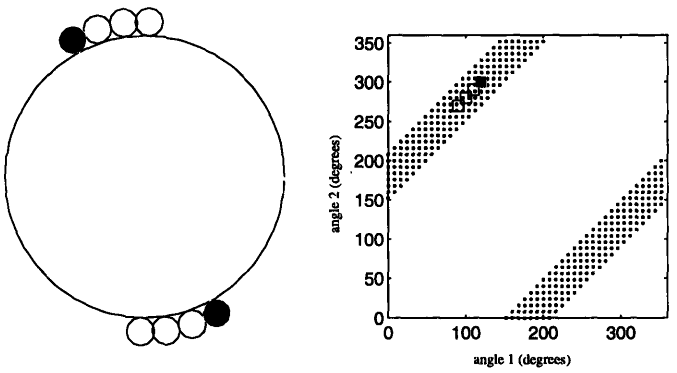

Figure 3.2 A two-fingered grasp which remains stable as both fingers increase in angle.

In fact, if we extend the grasp map beyond 360 degrees for both coordinates, we see that the stable region continues into the next map range. We refer to this larger map as an *extended grasp map*. (See figure 3.3.) Therefore this type of stable region is referred to as a *continuous region*. This is the type of region you would expect for a circle, since it is symmetric as it is rotated about its center. Any other change of finger positions, such as moving the fingers toward each other, would eventually result in an unstable grasp. (See figure 3.4.)

In contrast, consider an ellipse with an eccentricity of 100. The object and its grasp map, using the same coefficient of friction, is shown in figure 3.5. An initial grasp is shown both on the on the object and on the grasp map. In this case the grasp remains stable if the fingers move towards each other, shown in the grasp map of figure 3.6 as moving along a line with a slope of negative 1.

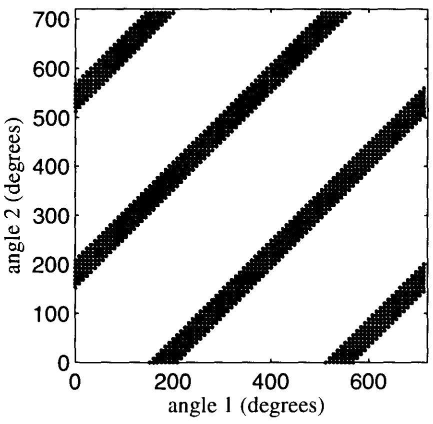

Figure 3.3 An extended grasp map for a circle, showing the continuous stable region.

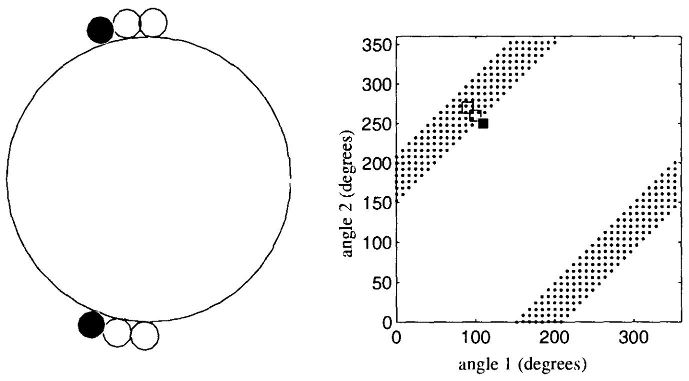

Figure 3.4 A two-fingered grasp which becomes unstable as one finger increases in angle while the other increases, bringing the fingers closer together.

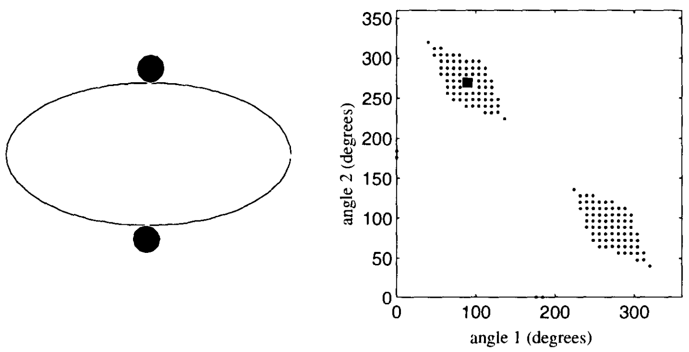

Figure 3.5 A two-fingered grasp of an ellipse and its associated grasp map.

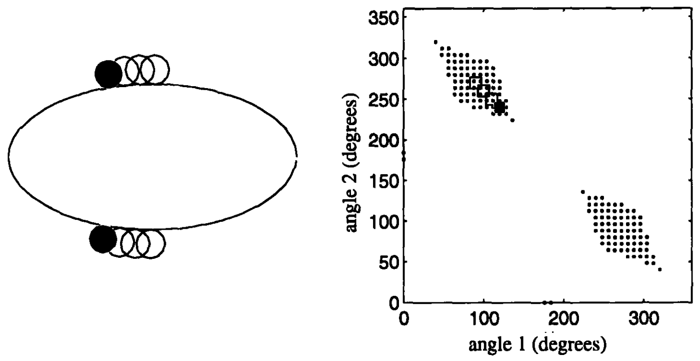

Figure 3.6 A two-fingered grasp which remains stable as the two fingers move towards each other.

Eventually, however, this change of grasp, as well as all others, will result in an unstable grasp. We call this a discontinuous region. There are smaller discontinuous regions in this map as well, which are more clearly evident in the object's extended grasp map, shown in figure 3.7. The shaded region at the coordinates (180,360) is one example. In these small regions, the fingers can shift very little before the grasp becomes unstable.

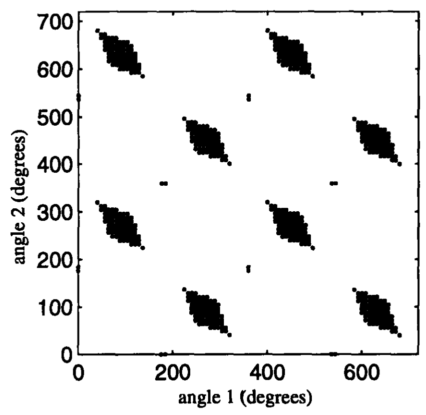

Figure 3.7 An extended grasp map for an ellipse showing the discontinuous stable regions.

In figure 3.8 we show the transition from +1 to -1 being the predominant stable region as we vary object eccentricity. We show this transition for various coefficients of friction in figure 3.9. As one would expect, the stable grasp regions shrink as the coefficient of friction decreases. The relevance of the alignment of the stable grasp regions to grasp gaits will be expanded on in Chapters 3, 5 and 6.

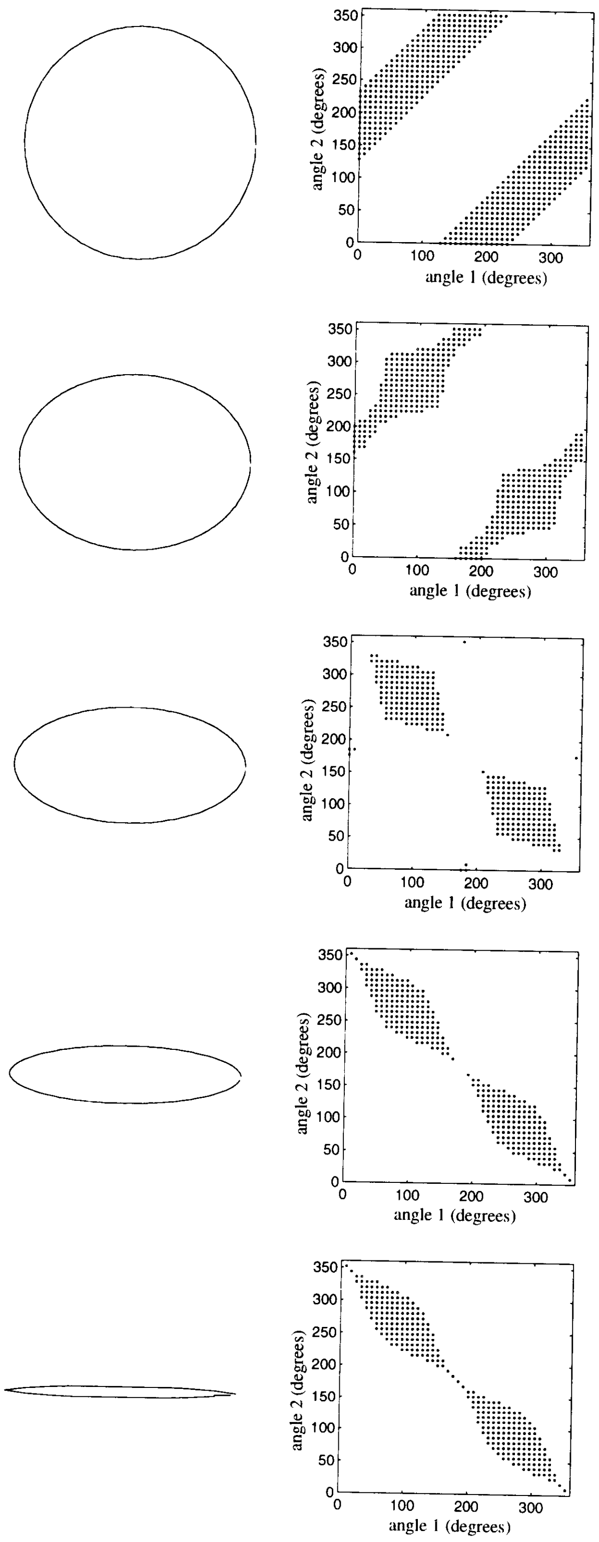

Figure 3.8 Objects of varying eccentricity with their associated grasp maps for a coefficient of friction = .5.

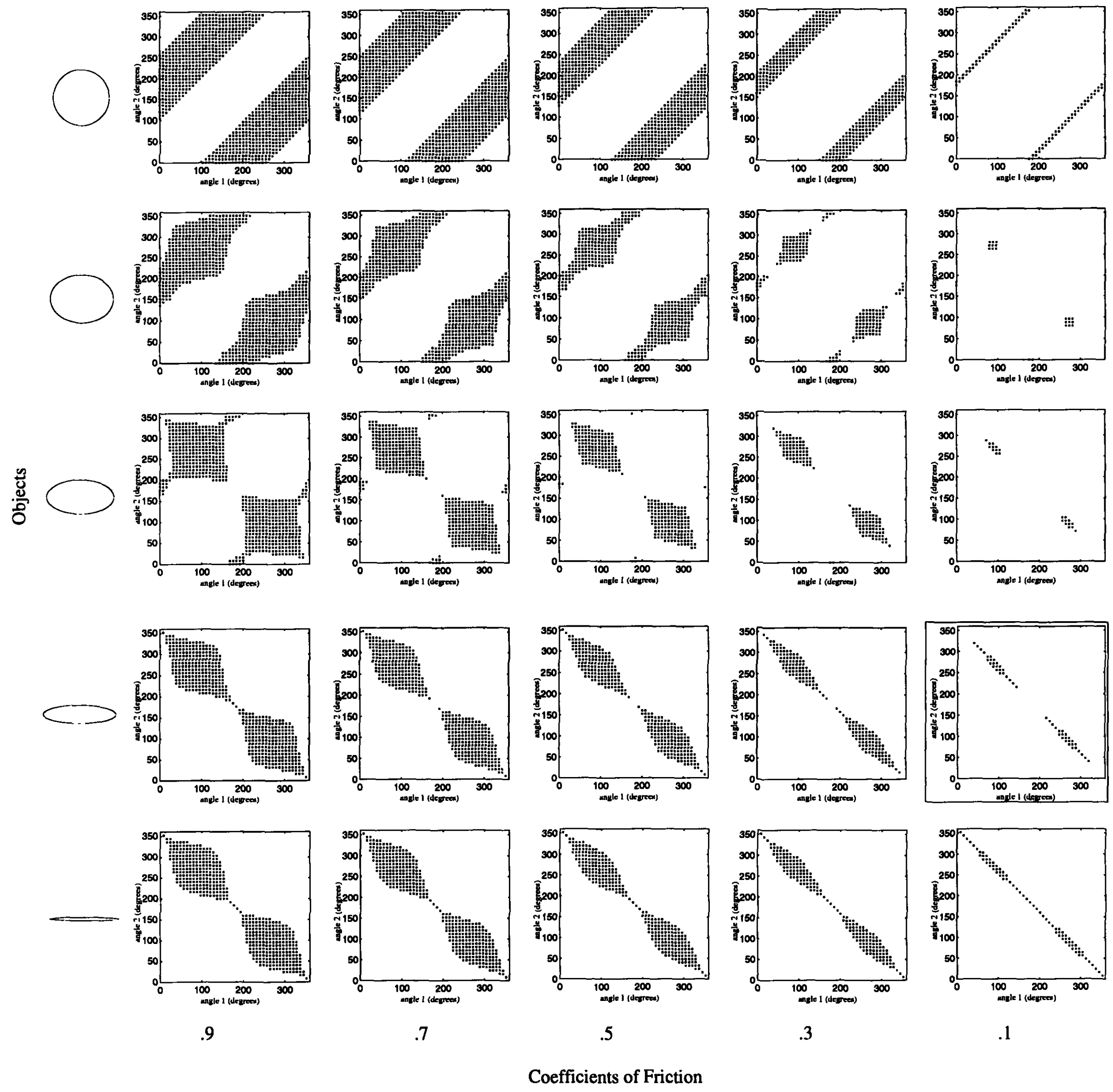

Figure 3.9 Grasp maps for the objects in figure 3.8 (eccentricity decreasing by row, starting from the top) for decreasing coefficients of friction (friction decreasing by column starting from the left: .9, .7, .5, .3, .1).

3.4 Object Smoothness and The Grasp Map

The second parameter which we use to characterize object shape is its smoothness. We define its smoothness *S* to be equal to the number of sides of the object. A perfectly smooth object (no discontinuities in slope) is defined to have infinite smoothness. Hybrid shapes are also possible, where part of the contour is smooth and part consists of line segments of constant curvature. In the case of hybrid shapes, the smoothness can be defined over sections of the contour. To show the effect of smoothness on the grasp map, we vary object smoothness while keeping the coefficient of friction constant and the eccentricity as constant as possible. If we begin with a circle, as smoothness decreases the eccentricity will have to increase, since all radii will not remain equal. However, keeping the eccentricity as constant as possible, grasp maps for objects with decreasing smoothness using a coefficient of friction of .5 are shown in figure 3.10. Figure 3.11 shows these transitions for varying coefficients of friction. We see that as smoothness decreases, the continuous stable region (along the +1 line) begins to break up into smaller, discontinuous regions. Each discontinuous region begins to elongate in the -1 direction. However, the regions themselves are still distributed along the +1 line for these low eccentricity values. This is because for a single grasp of these types of objects, where the fingers are on surfaces with constant slope, the grasp remains stable locally as the fingers move towards or away from each other. However, globally new grasp regions are found as fingers both move around the object in the same direction. (See figure 3.12.)

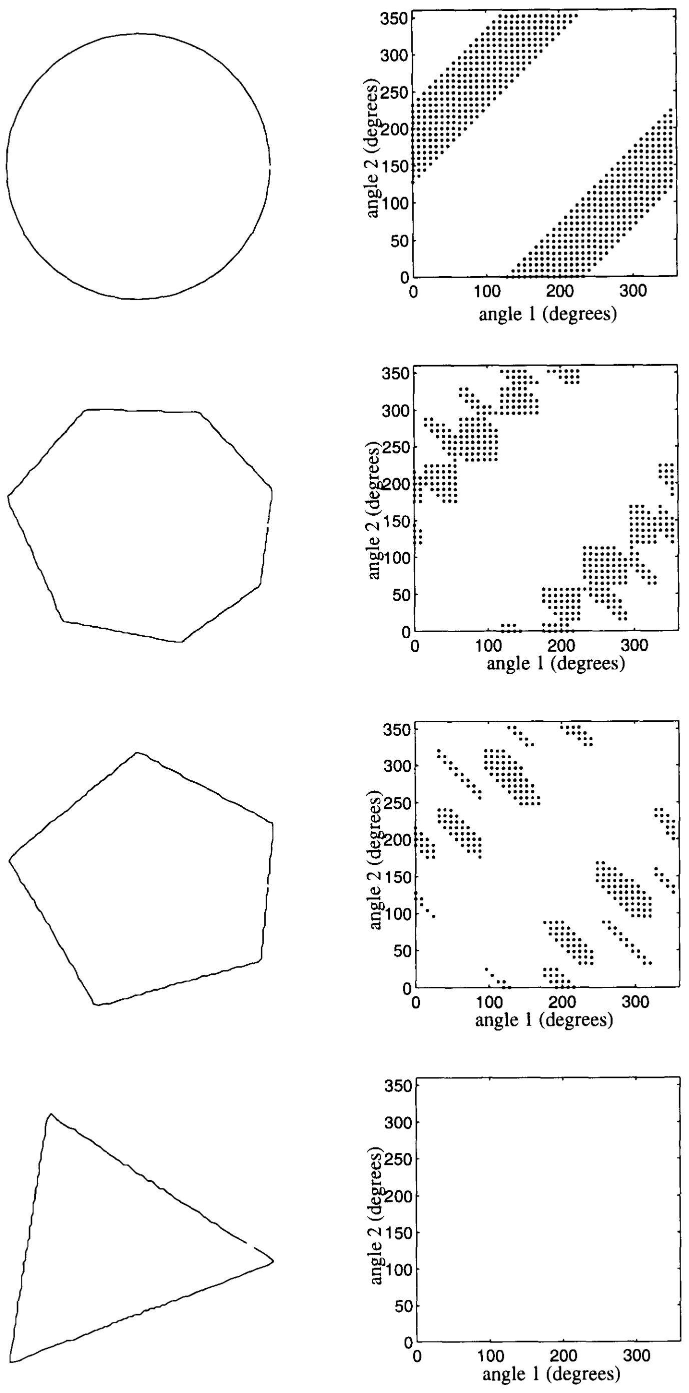

Figure 3.10 Objects of varying smoothness and their associated grasp maps for a coefficient of friction = .5.

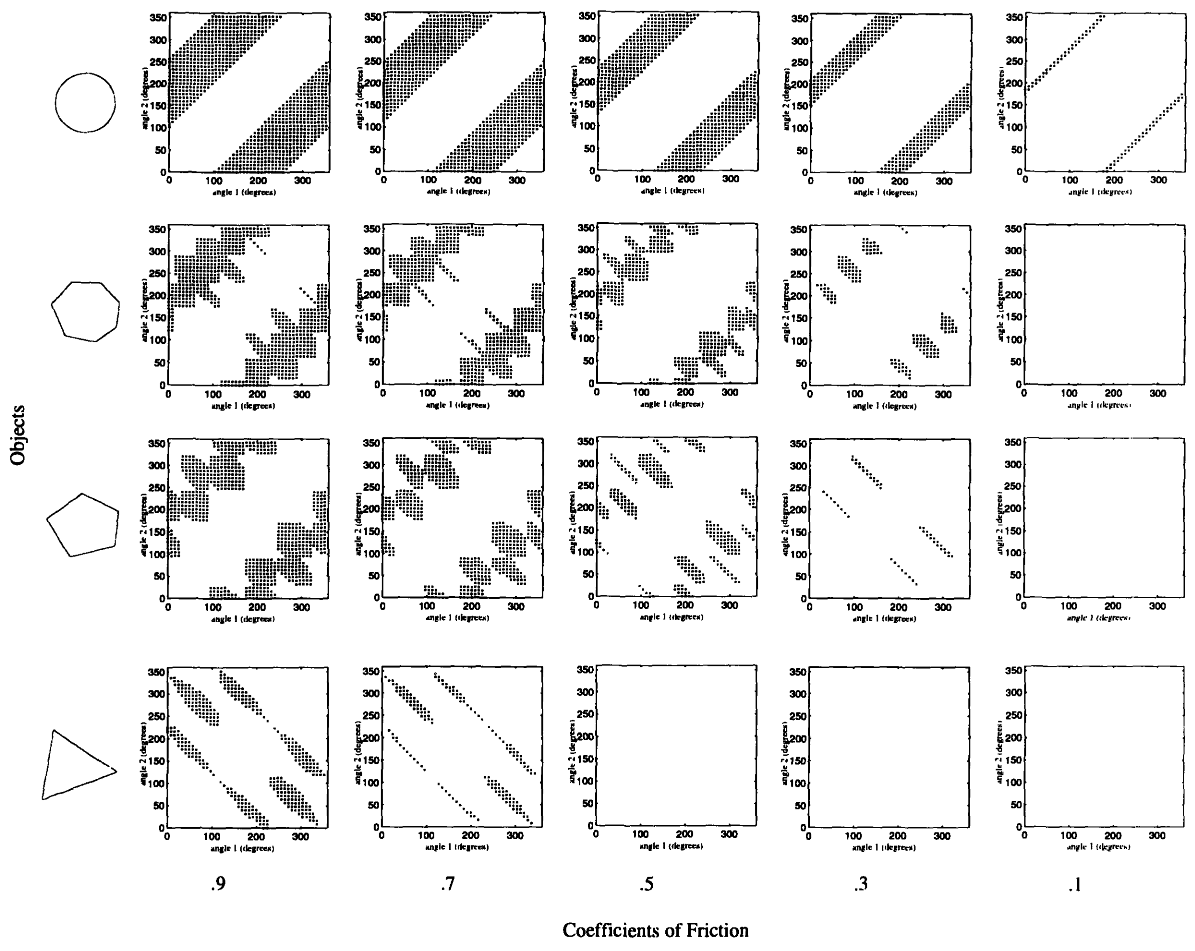

Figure 3.11 Grasp maps for the objects in figure 3.10 (smoothness decreasing by row, starting from the top) for decreasing coefficients of friction (friction decreasing by column starting from the left: .9, .7, .5, .3, .1). Note that for very low coefficients of friction, some objects have no stable grasps.

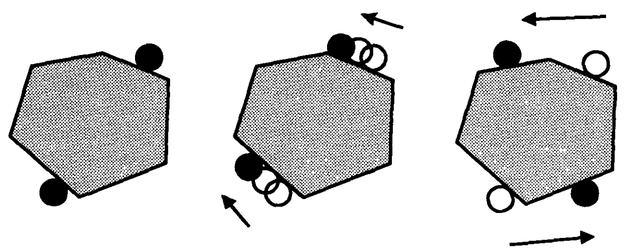

Figure 3.12 Series showing a two-fingered grasp for a pentagon, where locally the grasp remains stable if the two fingers move towards each other but globally a new stable grasp is found when both fingers increase in angle.

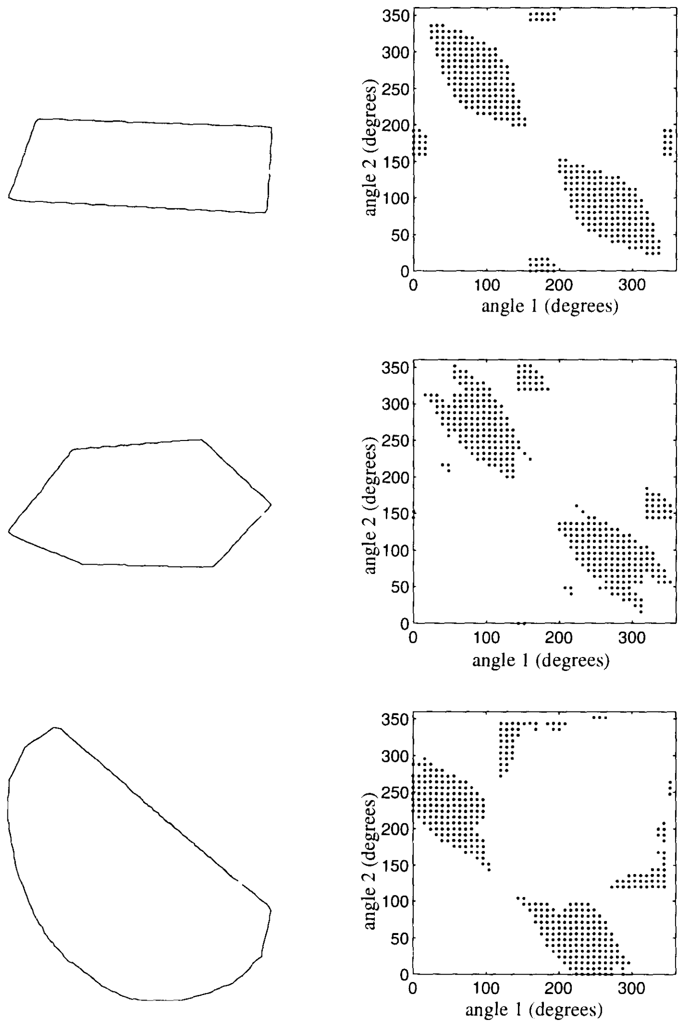

Figure 3.13 Various objects and their associated grasp maps.

Chapter 4

Workspace Limits and Grasp Gaits

4.1 Overview

When reorienting an object using any hand (human or robot), only a finite amount of motion can be imparted to the object before a new grasp must be found. The reason for this is that the fingers can move the object only a finite amount before they reach a limit to their range of motion. This limit may be due to an actual finger workspace limit or it may be due to an effective limit, caused by one of the links of the finger colliding with the object or with another finger. In section 4.2 a *workspace map* is constructed to represent the physically realizable two-fingered grasps for a robot hand with given its workspace limits. In section 4.3 the workspace map is combined with the grasp map to represent object motions, regrasps and grasp gaits. In section 4.4 I three examples of grasp gaits are presented using this representation.

4.2 Workspace Limits and the Workspace Map

For any robot hand, each robot finger has real workspace limits, beyond which the finger cannot move. When manipulating an object, each finger may, in addition, have *effective* workspace limits, where finger motion is further limited due to the presence of the object or other fingers. In this thesis a finger's workspace is defined by the range of angles through which it can move, measured in a reference frame *F*. Reference frame *F* is fixed to ground and shares its origin with reference frame *f*, the object reference frame. The finger workspaces are assumed to be equal in size and separation. If finger workspaces overlap, a finger cannot move to any place in its workspace at any time; rather where it can move is dependent on the state of the other fingers. Therefore, we have chosen the workspaces to be non-overlapping to avoid finger crossover problems. The space *between* the workspaces, where no finger can reach, becomes an important dimension in determining whether or not a given object can be reoriented. We call this dimension *w* and use it extensively in chapter 6. Figure 4.1 shows sample workspaces for fingers that can each move through a 90 degree range. The shaded regions represent areas where no finger can reach.

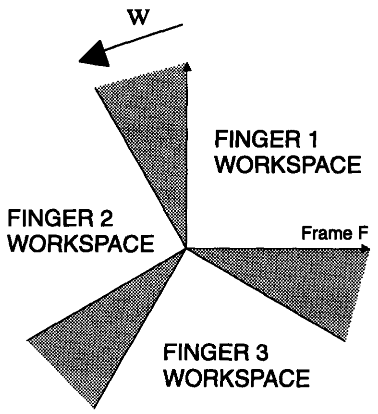

Figure 4.1 Finger workspaces defined from reference frame *F*.

Of course no actual finger will be able to reach any radius given its defined workspace. However for a given object, one can consider the range of angles through which each finger can move, such that it could contact the object at that angle for any object orientation. This range can be found by finding the angles of the two intersection points between the smallest radius of the object and the workspace as the object is rotated 360 degrees, as well as the angles of the two intersection points between the largest radius and the workspace. The more conservative pair of angles would then define the finger's effective workspace, since no radii within that pair of angles would lie outside of the effective workspace. An example of determining the effective workspaces of fingers with circular workspaces used to reorient an ellipse is shown in figure 4.2. Figure 4.2 shows an intersection point of the smallest radius on an ellipse with workspace 1. Also shown is the workspace intersection with the largest radius. Using the conservative workspace limit, which is the small radius intersection, the effective workspaces for all three fingers is shown.

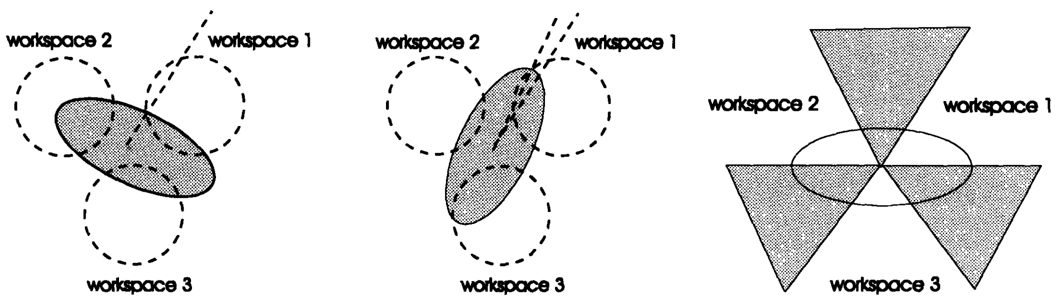

Figure 4.2 A series showing the construction of the effective workspaces of fingers given actual workspaces and an object shape.

For a large object, part of its contour may extend beyond the workspace while adjacent contour sections remain within the workspace. For the case shown in figure 4.3, the end of the ellipse is shown to extend beyond workspace 1. However, the effective workspace for each finger can be still defined using the method described above by choosing a smaller radius as the maximum radius. Then any point on the contour of the object which intersects the actual workspace of the object at an angle outside of the defined effective workspace cannot be used in a stable grasp. This can be ensured by considering any grasp which uses a contact at this point to be an unstable grasp and the grasp map for such an object can be modified accordingly. This method can also be used for any radii that still intersect but limit the effective workspace too significantly.

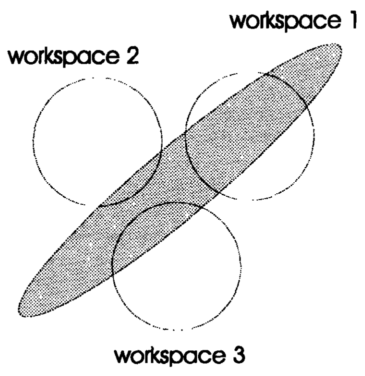

Figure 4.3 An object/workspace combination where part of object contour extends beyond the workspace boundary.

These workspace limits can now be represented in a two-dimensional space similar to the grasp map representation. The horizontal coordinate represents the angle of finger i with respect to frame F, and the vertical coordinate represents the angle of finger j with respect to the same frame. However, now the value of either coordinate implies a specific finger, since for every angle in reference frame F there is only one finger that can move to that position. In addition, there may be angles to which no finger can move. Therefore any point in this space corresponds to either a grasp of a particular pair of fingers, or a grasp which cannot be achieved due to the specific workspace limits. A workspace map can be constructed by defining the boundaries of regions which contain valid finger pairings. These boundaries are squares for the symmetric and non-overlapping workspace case. We call these *finger pair boxes*. For a grasp to satisfy workspace constraints, it must be within one of the finger pair boxes. For example, using the workspace limits shown in figure 4.1, where finger 1 can move between 0° and 90° , finger 2 between 120° and 210° and finger 3 between 240° and 330°, the open box in the top left hand corner of figure 4.4 corresponds to grasps between fingers 1 and 3. Again, the workspace map is symmetric about the $y=x$ line, and it repeats every 360° . We refer to the finger pair boxes that are furthest from the $y=x$ line as *outer* boxes, and those closest to the $y=x$ line as *inner* boxes. If the workspace map is extended beyond 360°, the inner and outer boxes are bound by two lines with a slope of positive 1. The vertical and horizontal spaces between the finger pair boxes are equal to w, the space between the finger workspaces.

Using the workspace map, we then look for grasp gaits whose regrasps occur according to the following rules. First, it is assumed that when switching from one two-fingered grasp to another, there is always one finger and its corresponding position which is common to the two grasps. (It is assumed that the new finger is placed on the object before the old finger is removed, so that there are always at least two fingers on the object.) Therefore, to switch to a new grasp and corresponding point on the map, only vertical or horizontal moves are permitted, since these are the only moves which change only one finger at a time. Further, a new grasp cannot be found within the same finger pair box, since this would require that the same finger be removed from the object and replaced at another position, during which time there would be only one finger remaining on the object.

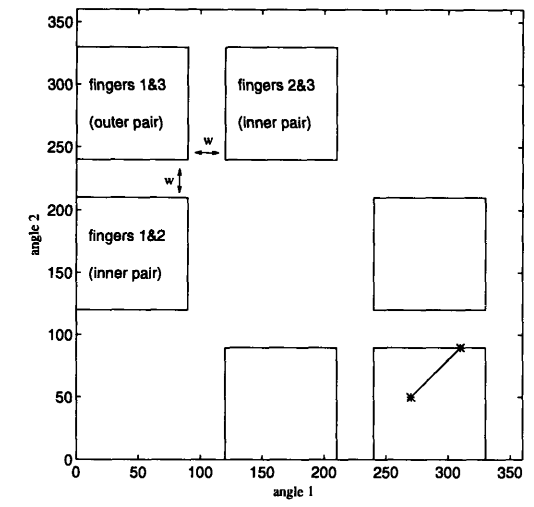

Figure 4.4 A workspace map for workspaces shown in figure 4.1.

When the object is rotated about the origin of both reference frames f and F, both of the fingers in the grasp increase their angles by the same amount as seen in reference frame F. Therefore, in the workspace map, rotations are represented by the grasp point moving along a diagonal line with a slope of 1. For counterclockwise rotations, both finger positions increase, and for clockwise rotations, both decrease. If the object continues to rotate, eventually the grasp point will hit one of the edges of the finger pair box, corresponding to one of the fingers hitting an edge of its workspace. In figure 4.4, a two fingered grasp defined by the finger positions (270, 50) is represented by the lower left hand star in the lower right hand finger pair box. The line traces the change of coordinates with respect to reference frame F as the object is rotated 40° counterclockwise. The upper right hand star represents the final finger positions in workspace frame of (310,90) and shows that the object can no longer be rotated using such a grasp.

4.3 Combining the Maps to Represent Grasp Gaits

Since reference frames F and f share a common origin, before the object is rotated the two frames are coincident and thus the two maps can be overlayed. (See Figure 4.5 (a).)

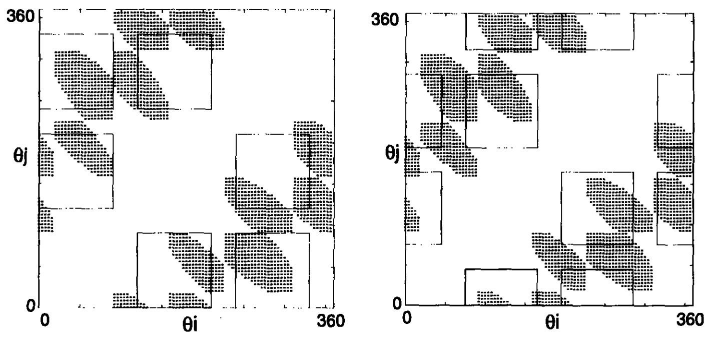

Figure 4.5 Series showing the effect of object rotation on the grasp and workspace maps.

Feasible grasps can now be seen to be those which satisfy both the grasp stability and workspace constraints. When looking for a new grasp, the new grasp point must be in a region shaded in the grasp map, and it must also be either a vertical or horizontal move from the original point, as well as in a new finger pair box. As the object is rotated, not only the point representing the present grasp travels along a diagonal line as seen from the workspace map, but every point in the grasp map travels along a diagonal line, since every point on the object rotates by the same amount relative to the workspace reference frame. Therefore a rotation of the object is represented by a sliding of the grasp map along the diagonal of the workspace map. For a counterclockwise rotation, the grasp map would slide up and to the right relative to the workspace map; for a clockwise rotation, it would slide down and to the left. In figure 4.5 (b) we show a counterclockwise rotation of 45° as seen from the object frame. The finger pair boxes have slid along the diagonal in the negative direction, from the upper right corner to the lower left. Since the workspace map is repeating, the box edges that have disappeared from the lower left corner have reappeared in the upper right corner.

The problem of finding a gait can now be posed in map space. The goal of rotating an object continually is equivalent to the goal of continually sliding the finger pair boxes of the workspace map along the positive diagonal of the grasp map. The constraint that the object must be held at all times by a stable two fingered grasp is equivalent to the constraint that there must always be a point which is both in the stable region of the grasp map and inside one of the moving finger pair boxes of the workspace map. The problem of finding a gait is equivalent to finding a series of points (1) which satisfy the above constraints, (2) which are connected in grasp map space by only vertical or horizontal moves, and (3) where no two consecutive points are in the same finger pair box in workspace map space.

4.4 Some Example Grasp Gaits

In looking at gaits which successfully rotate objects, we note that for certain objects and grasps, the object can be rotated until a finger reaches the end of a workspace, at which point a new grasp can be found which will allow rotation to continue. Figure 4.6 (1)-(4) shows a sequence of rotations and regrasps for a polygon with its representation in grasp map space to the right. Figure 4.6 (1) shows an initial grasp of (50,270) in grasp map coordinates, between fingers 1 and 3, represented by the *x* in the outer finger pair box in map space. Figure 4.6 (2) shows the object after being rotated counterclockwise by 40°. Finger 1 has reached the end of its workspace and thus the object can no longer be rotated. Figure 4.6 (3) shows that a grasp between finger 1 and 3 has been replaced by a grasp between fingers 2 and 3, which is a positive horizontal move in the grasp map space to the coordinates (140,270). Figure 4.6 (4) shows the state after the object has been rotated an additional 20°. Looking at map space, we see that a positive vertical move in map space, equivalent to a regrasp by the fingers, will allow the object to be rotated further.

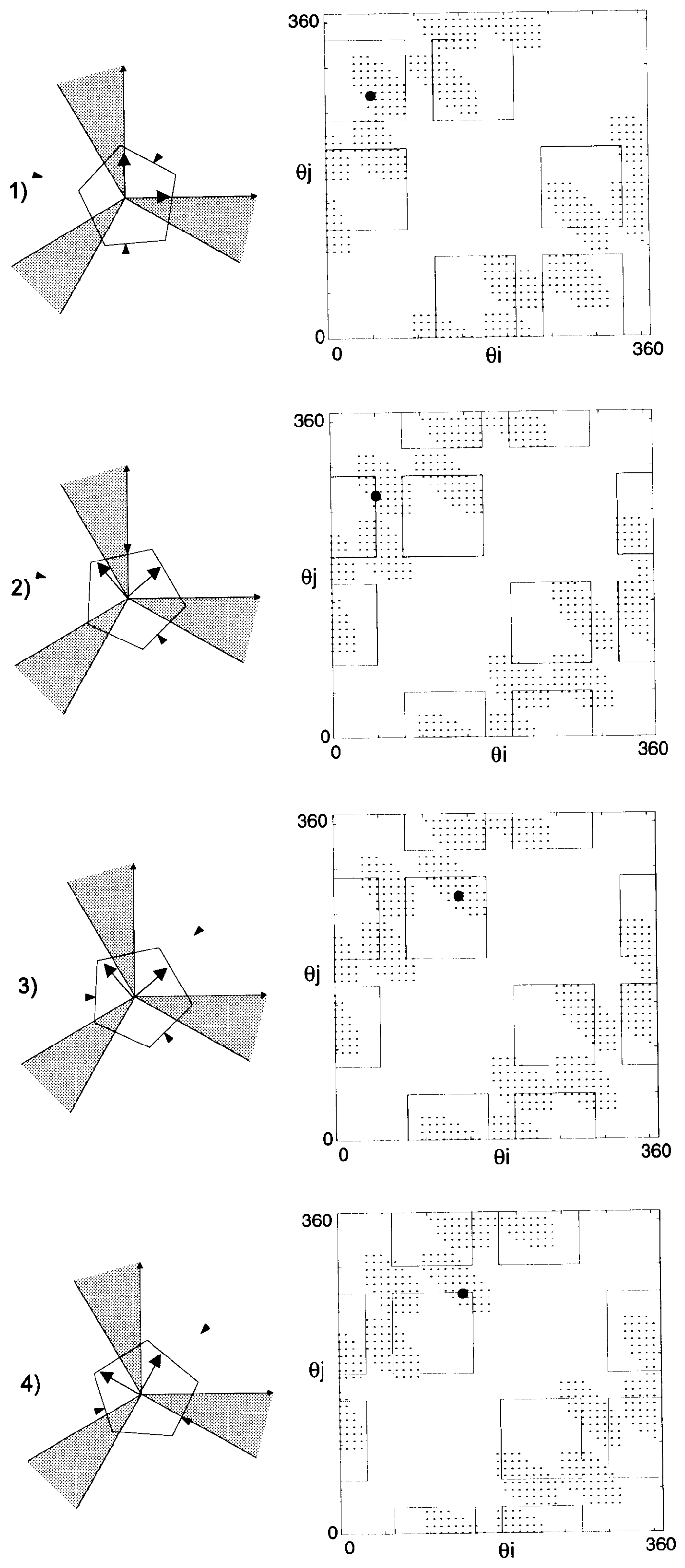

Figure 4.6 Series showing pentagon rotating successfully.

For other objects or other grasps, a new grasp cannot always be found if the object is rotated until a finger reaches a workspace limit. However, in some cases, a new grasp *could* have been found if it was sought at an earlier point in the rotation. Figure 4.7 and 4.8 illustrate this point. Figure 4.7 (1) shows an ellipse with an initial grasp of (135,245) in grasp map coordinates, using fingers 2 and 3. Figure 4.7 (2) shows the ellipse after it is rotated counterclockwise by 75°. Finger 2 has reached the end of its workspace. The corresponding map shows that although a vertical move in the map is possible, the second finger will still be at the edge of its workspace, and no additional moves to a stable region are possible which at the same time will allow the object to be rotated further. Figure 4.8 (1)-(4) shows an alternative path that will allow the object to be rotated beyond this point. Figure 4.8 (1) shows the object after it has been rotated only 45° from its original position. Figure 4.8 (2) shows a negative horizontal move in the grasp map to a grasp of (45,245) in grasp map coordinates. Figure 4.8 (3) shows a subsequent positive horizontal move to a grasp of (75,245) which now enables the object to be rotated further. Figure 4.8 (4) shows the object after being rotated to a position of 85°, at which point finger 3 has reached the end of its workspace. The corresponding map shows that a positive vertical move will allow the object to be rotated further still.

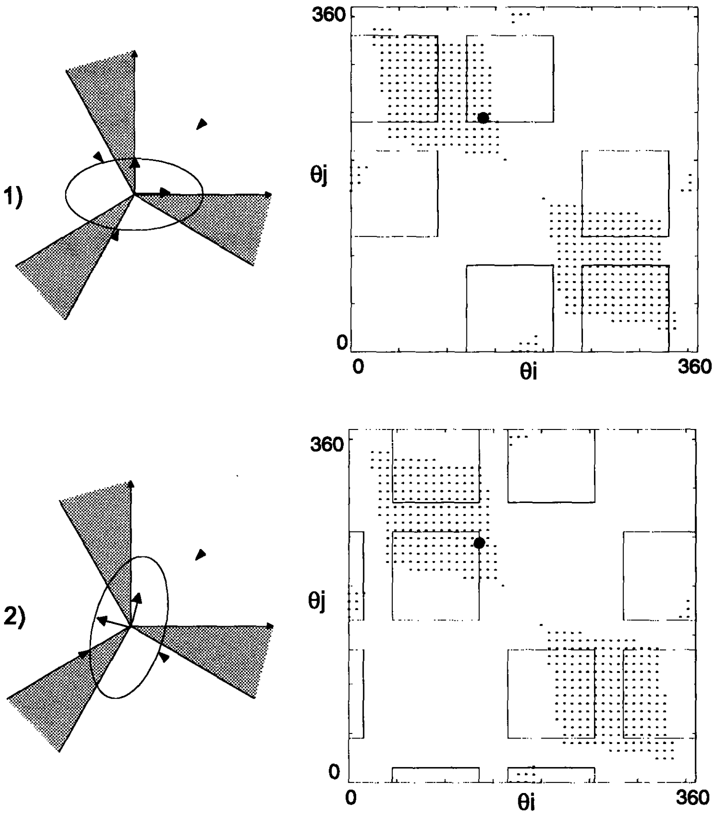

Figure 4.7 Series showing ellipse unable to rotate further.

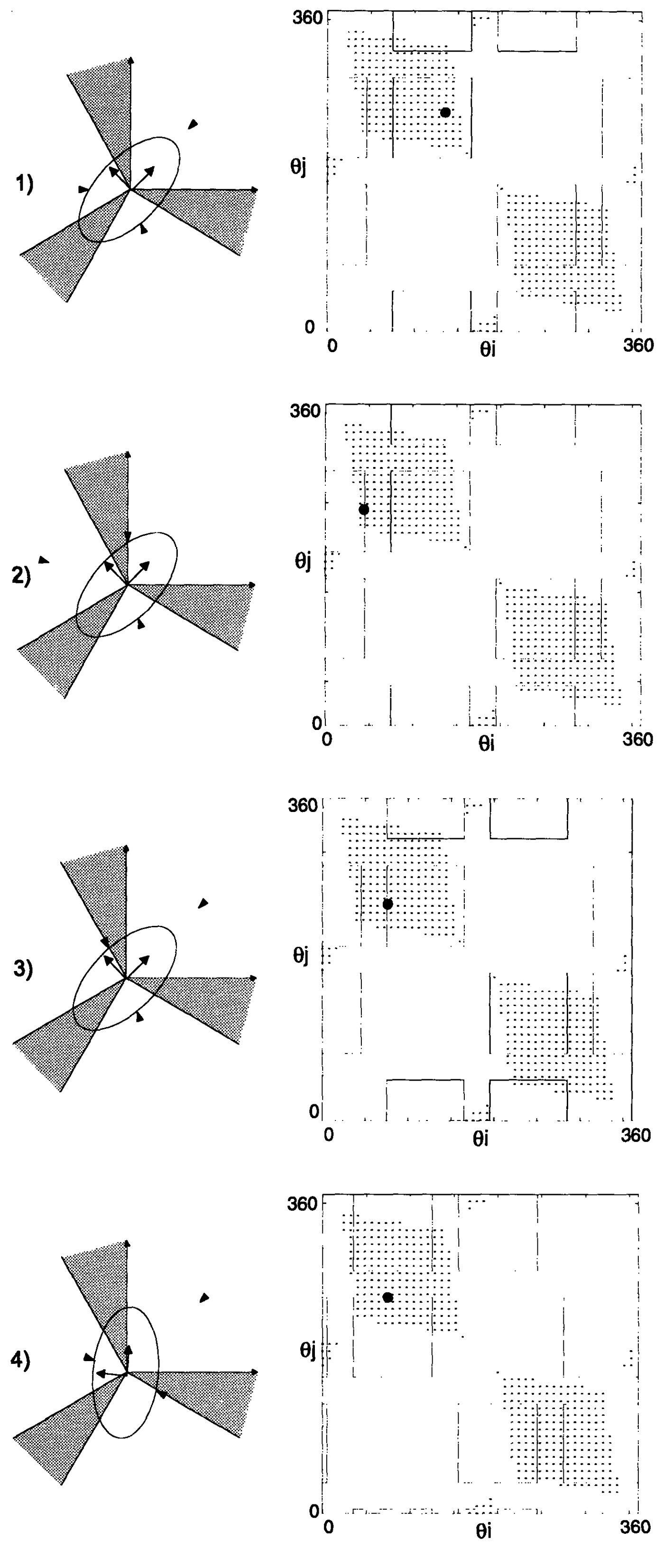

Figure 4.8 Series showing ellipse rotating successfully.

Chapter 5

Searching For Gaits

5.1 Overview

This chapter presents a variety of search techniques constructed to find a sufficient gait to reorient a given object 360°. In section 5.2 the search problem is formulated. Section 5.3 discusses breadth first searches for this problem. Section 5.4 discusses depth first searches. Section 5.4 develops a metric in order to guide the search, using a best first method and presents the results of this search for several different object shapes. While the method described in section 5.4 produces good results, section 5.5 motivates the need to look further into the problem of generating gaits.

5.2 Brute Force Search

One way to look for a gait that could successfully rotate an object is to do a brute force search through all of the possible sequences of motions. Any step in a gait, or state of the system, can be completely described by three variables: the angle of the object relative to reference frame *F* (the workspace reference frame) and the angle of the two fingers grasping the object relative to reference frame *f* (the object reference frame). Then for any state, all the possible actions which can occur next can be enumerated using the grasp map and workspace map. At a given state two actions are possible: rotating the object or regrasping the object. The object can be rotated by an angle just greater than zero to an upper limit given by the minimum distance from the grasp point to the right or upper edge of the finger pair box in the workspace map. (See figure 5.1.)

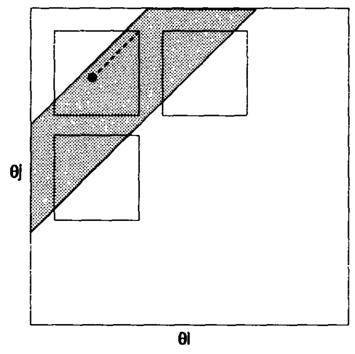

Figure 5.1 Given the grasp shown by the solid dot, the range of possible object rotations is shown by the dotted line.

This range of possible object rotations is discretized by Δθ to allow searching through discrete choices. Alternatively, the present grasp can be replaced by another grasp, by moving horizontally or vertically to a new finger pair box in the workspace map to any point in a shaded region in the grasp map. (See figure 5.2.) These two ranges of moves within the grasp map are discretized by Δx and delta Δy.

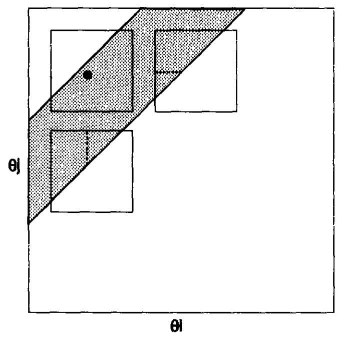

Figure 5.2 Given the grasp shown by the solid dot, the range of possible new grasps is shown by the dotted lines.

The number of alternative moves from each state depends on the state, the object shape, the coefficient of friction and the discretization chosen for the object and finger angles. A search tree can be constructed, where each node in the tree represents a state in the gait, $\underline{X}$, where

$$\underline{X} = \begin{bmatrix} \theta \\ x \\ y \end{bmatrix}.$$

θ is the object angle as measured from the workspace frame, x is the horizontal coordinate in the grasp map, and y is the vertical coordinate in the grasp map. The specific finger pair used in the particular two-fingered grasp can be determined from the state and from the knowledge of the workspace limits. From each node emanates a branch, representing an action, leading to another node or state. All possible actions which can occur at a given state must be included. Generally when search trees are made for path planning, each variable contributes only two branches to a given node, the action associated with the branch being $x = x+\Delta x$ and $x = x-\Delta x$. This is due to the fact that every other state involving an x greater or less than these would have to pass through one of these states in order to get to a different one. For gait planning, however, a finger position can be thought of as a *discontinuous* variable, in the sense that it can move to any number of neighboring contact positions without having to pass through the previous ones. Therefore, the branches corresponding to changing the x and y coordinates leaving from each node must include *all* possible final positions for each finger. Thus, for gait planning, the number of alternative motions, or the branching factor of the search, is large. Using a discretization of 2° for both finger and object positions yields a typical branching factor of 35.

5.3 Breadth First Search

A breadth first search explores every possible action from a given grasp as a possible path before exploring the next level of possible motions in the search tree. Therefore, a breadth first search would be guaranteed to find the shortest gait. However, with branching factors on the order of 35, a breadth first search would be expected to be unfeasible due to computer memory limits. We ran a breadth first search for a number of different objects, with a goal of rotating the object 360°. For none of the cases was the search anywhere near a solution after using 500Mb of disk swap space. Thus a breadth first search was considered intractable.

5.4 Depth First Search

Alternatively, a depth first search was tried for a variety of objects. A depth first search explores only one possible motion from a given grasp until either the goal is met or a dead end is reached. If a dead end is reached, the search continues from a possible motion from the closest previous node. Since object rotation is the goal of the search, the nodes which involved rotation were put at the head of the queue, in front of the nodes involving regrasps. Therefore, regrasps were only sought if rotation was no longer possible. Using this approach, a solution can be found if a solution exists, but the paths that are found can be extremely long and therefore inefficient. For example, using a pentagon similar to the one shown in figure 4.6 and a coefficient of friction of .7, a path was found after creating over 2,000,000 nodes, expanding over 2,000 of them, and resulting in a path length of over 300. (Each node represents a state in the system defined by the particular grasp and object orientation. A path length is the number of nodes in the final path which leads to the goal.) Therefore, a depth first search was not considered to be efficient for finding gaits if they existed for general object shapes.

5.5 Best First Search

One of the reasons that a depth first search creates and expands so many nodes while trying to find a gait is that many regrasps are investigated which do not lead directly to continued rotation. A second reason is that the graph of possible paths is highly interconnected. (Since the number of partial paths leading to the same node can influence the number of times that each path leading away from that node is investigated, a highly interconnected graph may lead to a lot of redundant searching.) This second problem is easily solved by considering only the first path found leading to a given node and terminating all other paths leading to that same node. The first problem can be addressed by using a metric to evaluate each partial path and then choosing the node with the highest value to be expanded next. To avoid unnecessary regrasps, the value of the path should be decreased each time a new grasp is used. To guide the path towards rotation, the value of the path should be increased as the object is rotated. To avoid a rotation occuring as the sum of a number of smaller rotations, the value of the path is should also decreased when the object is rotated. Therefore, we choose a metric

$$M = \alpha\theta - \beta\eta - \gamma r,$$

where θ the object angle in degrees, η the number of regrasps, and r the number of rotations. A range of α, β, and γ were used and all of them equaling 1 gave satisfactory results. If $\alpha/\beta >> 1$, the object was often rotated too far and many unnecessary regrasps were tried before backing up to find an alternate path. If $\gamma = 0$, all combinations of rotations adding up to the maximum rotation was tried before backing up if a dead-end had been reached.

This *best first* approach was used to search for a gait to rotate many different object shapes by 360°. In every case the search was able to determine whether or not a path existed. Figure 5.3 shows 4 objects for which successful gaits existed using a workspace size for each finger of 90 degrees. In each case, 10 different initial grasps were used. Table 1 shows the results for the best first search for a coefficient of friction of .8 and .7. Three different data types were used as measures to compare results: average *nodes opened*, average *path length* and average number of regrasps required by the final path. *Nodes opened* are nodes which have been explored as a possible path. (The number of nodes created is generally several orders of magnitude greater than the number opened for this problem.) The *path length* includes both regrasps and rotations, since a state change includes either a regrasp or a rotation. The difference between the number of nodes expanded and the length of the path can be used as a measure of the efficiency of the search, since the number of nodes expanded that were not in the final path led to dead ends. In chapter 6, these results will be considered again when comparing rule-based methods for gait planning to search methods.

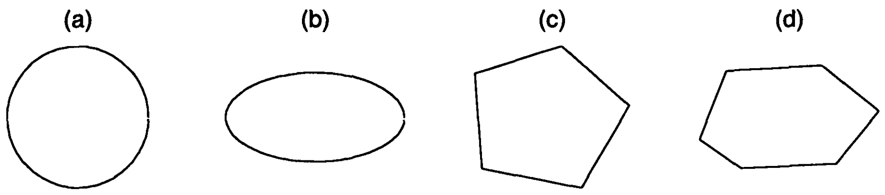

Figure 5.3 Objects used for testing best-first search method for generating gaits.

DATA TYPE	circle (a)	
	μ=.8	μ=.7
nodes opened	500.2	957.7
path length	44.8	82.0
regrasps	22.4	41.1
	ellipse (b)	
	μ=.8	μ=.7
nodes opened	5,746.5	14,486.3
path length	48.2	30.4
regrasps	31.5	20.3
	pentagon (c)	
	μ=.8	μ=.7
nodes opened	315.0	208.2
path length	37.4	29.5
regrasps	21.9	18.0
	hexagon (d)	
	μ=.8	μ=.7
nodes opened	439.5	1,421.2
path length	42.7	83.4
regrasps	24.4	58.0

Table 1: Results of best first search for objects shown in figure 5.4 for coefficients of friction equal to .8 and .7.

5.6 Still More Questions

Thus, using the tools developed in chapters 3 and 4, along with a metric to direct the search toward a reasonable path, we were able to construct a search which finds sufficient gaits to rotate objects. However, finding a sufficient solution using a search method does not in itself lead to an understanding of the grasp gait problem. There are still questions we would like to answer. Why does a gait exist in some cases and not in others? Why is a gait harder to find in some cases than in others? Is there an instructive way to classify different types of gaits? Answers to these questions would be interesting in their own right as well as helpful in finding more efficient solutions than the best first search. In addition, they would be helpful in guiding methods to manipulate objects when only general information about the object is known, and a robot hand with sensors is searching for the next stable grasp instead of a computer, in which case maximum efficiency is essential. To answer these questions, we take a closer look at a few of the grasp maps and the type of gaits that can be generated.

Chapter 6

Finding Prototype Gaits

6.1 Overview

As shown in chapter 5, gaits can be found by searching through a tree of the possible finger positions and object rotations until the goal position is found. When this search is guided by a metric that favors paths with fewer steps and larger object rotations, the gaits are of reasonable length. However, there is a tradeoff between how long it takes for a gait to be found and how efficient the gait is in getting the object to its final position, and often a gait that takes hours to generate is still not very efficient. In addition, using a search provides no insight into why a certain gait was generated, how it is similar to other types of gaits, whether a gait will exist for a specific object, or what would need to be changed in order for a gait to exist. In this chapter two classes of gaits are defined, based on the types of moves that are made in grasp map space. The types of moves that are possible are based on the type of grasp map for a particular object, which is in turn based on global object properties. In section 6.2 grasp maps are classified into two different types by estimating the stable grasp regions as a simple shape that fits within the more complex individual map. In section 6.3 rule-based methods are developed which generate gaits for objects whose grasp map approximations satisfy certain necessary conditions. In section 6.4 these rules are used to prune the search tree for objects where the necessary conditions for the rule-based gaits are not met.

6.2 Finding Simply Shaped Regions in Complex Grasp Maps

In chapter 3 we considered the effect varying object eccentricity had on the grasp map. Below we show extended grasp maps for smooth objects of decreasing eccentricity, with extended workspace maps overlayed. We see from figure 6.1 that the grasp map for a circle lies along the same direction as the workspace pair boxes. Therefore we can see that at least for certain grasps it will not be difficult to find a new grasp once a workspace limit is reached. A new grasp can generally be found by moving forward in the grasp map.

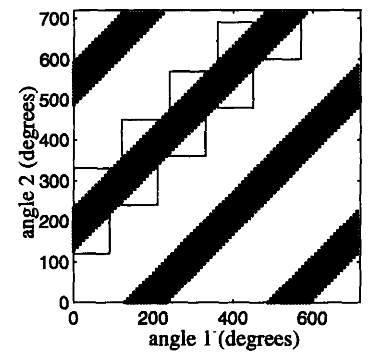

Figure 6.1 Extended grasp map for a circle with workspace map overlayed showing that the stable grasp region lies along the same direction as the workspace pair boxes.

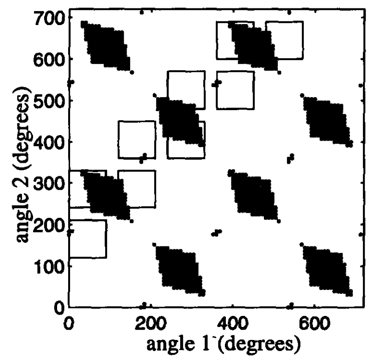

Figure 6.2 Extended grasp map for an ellipse with workspace map overlayed showing that the stable grasp regions do not lie along the same direction as the workspace pair boxes.

We see from figure 6.2, however, that for a more eccentric ellipse, the grasp map does not lie along the same direction as the workspace pair boxes. We see that for certain regions in the grasp map, a new grasp will not be found by moving forward in the grasp map when a workspace limit is found. In for this case, a new grasp must be found by moving backwards in the grasp map, and this can only happen at certain object orientations. (See figures 4.7 and 4.8.). Therefore we consider two types of gaits: one in which a new grasp is always found by moving forward in the grasp map, and one in which new grasps sometimes require moving backwards in the grasp map.

To study the problem further, we find two simple patterns within these maps and consider what makes a gait possible using just these regions. We also seek rule-based algorithms to generate the gaits so that searches are not necessary. Then any object that has one of the simple patterns within its map can be reoriented using those rules, since the gaits depend only on the grasp and workspace maps. For move-forward gaits we estimate the stable region of the grasp map to be a band of varying width and placement, lying along the positive diagonal, where the inner and outer edge have a slope of positive 1. This shape fits into many continuous grasp maps. Figure 6.3 (a) shows a example grasp map that includes the band prototype. For back-up gaits we estimate the stable region of the grasp map to be a box of varying height, width and placement. This shape fits within many discontinuous grasp maps. Figure 6.3 (b) shows an example of a grasp map that includes the box prototype. In the next section we will consider what the necessary conditions for these shapes are such that a gait is possible and what rules will generate these gaits.

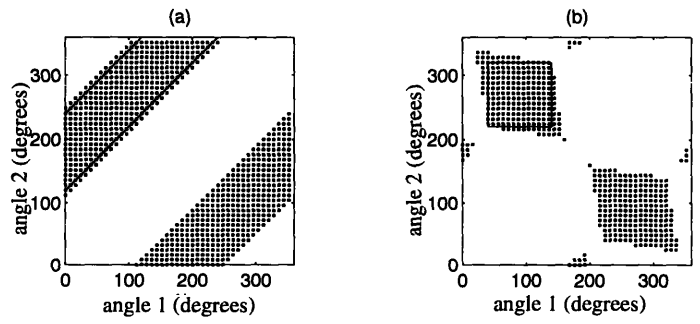

Figure 6.3 Prototype grasp regions and grasp maps.

6.3 Finding Prototype Gaits Using the Simply Shaped Regions

6.3.1 The Forward Gait

For many low eccentricity objects, a band readily fits within its more complex grasp map. We assume this band's inner and outer edge to have a slope of 1. Since object rotations are represented in map space by a relative sliding along the 45 degree line of the grasp and workspace maps, this band is invariant to object orientation as seen from the workspace map (See figure 6.4). Therefore the type of move that is possible is not dependent on the object orientation for the move-forward gait, although when the moves can occur may depend on orientation.

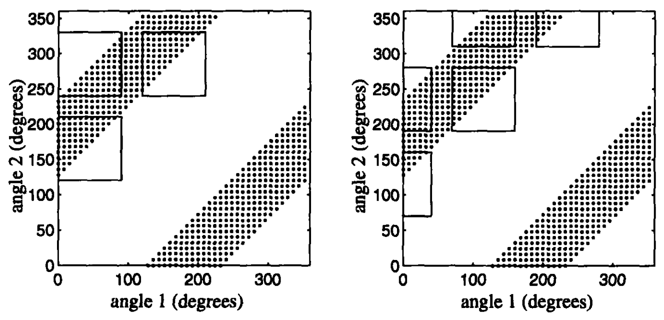

Figure 6.4 Series showing stable grasp region for band is invariant to object angle.

To ensure that a new grasp can always be found within the band after rotation, we will consider the constraints that must be met, assuming counterclockwise rotation within an inner workspace box. (Recall that counterclockwise rotation is represented by each point in the grasp map moving along a 45° line, up and to the right, as seen from the workspace map.) From symmetry, the same constraints hold for an outer workspace box and for clockwise rotation as well. First, we note that the minimum size step that can be taken in a workspace map occurs when a finger has reached a workspace limit, which occurs when the grasp hits either the upper workspace edge or the right workspace edge. (See figure 6.5.) If the grasp is such that it hits the right workspace edge, the requirement that the band intersects both workspace pair boxes will ensure that a new stable grasp (lying within the band) will exist in the upper box. This is condition 1. (See figure 6.6.) If the grasp is such that it hits the upper workspace edge, a new grasp will not be found unless the band intersects both workspace boxes and has a height of at least w, since a step of at least w (the space between the workspaces as defined in chapter 4) is required due the workspace map constraint. This is condition 2. (See figure 6.7.) In addition, the *location* of the grasp in relation to the grasp map is relevant for this case. In particular, the grasp must be at least a distance w from the forward edge of the band, where the forward edge is the edge towards which the new grasp would move. This is condition 3. (See figure 6.8.)

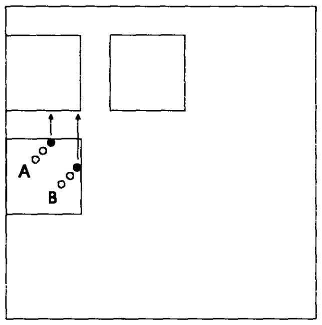

Figure 6.5 Figure showing that the smallest step taken to a new grasp occurs when a finger reaches a workspace edge. Grasp A and B move from the open circle to the filled circle as the object is rotated. The smallest step to the next workspace pair box occurs when either grasp reaches a workspace limit.

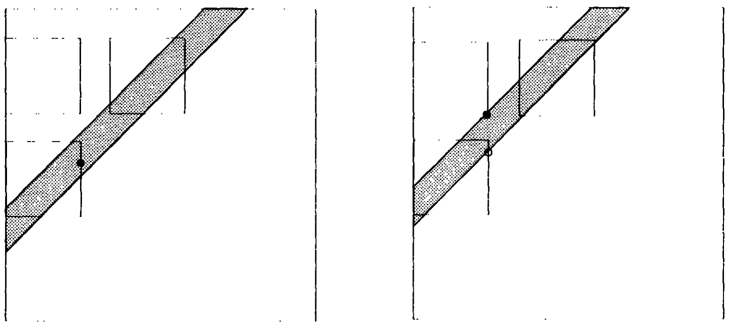

Figure 6.6 Series showing that the band must intersect both inner and outer workspace pair boxes if a new grasp is to be found when a grasp reaches its right workspace edge within an inner pair box.

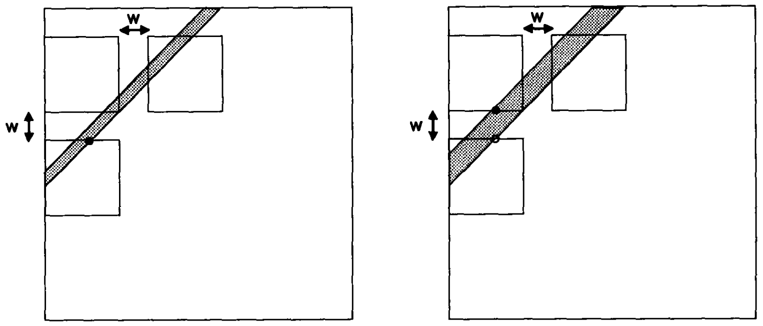

Figure 6.7 Series showing that the band must have a height of at least *w* if a new grasp is to be found when a grasp reaches its top workspace edge within an inner pair box.

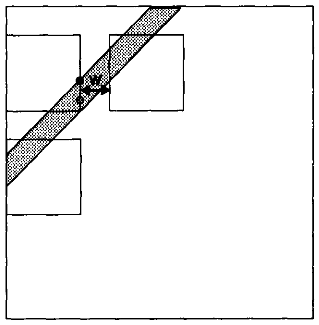

Figure 6.8 Figure showing that the grasp must be *w* from the forward edge of the band if a new grasp is to be found when a grasp reaches a workspace edge.

Therefore, an object whose grasp map contains a band which satisfies the above 3 conditions can be rotated continuously for the following reasons. The 3 conditions ensure that the first new grasp can be found once a finger reaches a workspace edge. The first 2 conditions are invariant with respect to grasp changes, so that they will be true for all following grasps as well. The third condition remains true if it was true for the first grasp, since any new step forward must be at least of size *w*, leaving a distance of at least *w* between the new grasp and the next forward edge. (See figure 6.9.)

Therefore, an object can be rotated continuously by rotating until a finger reaches the end of a workspace and then switching to a new grasp that allows rotation to continue. In fact, the object does not have to be rotated until a finger reaches the end of its workspace, but it needs to be rotated until the distance the grasp is from the edge it is going toward in grasp map space is equal to or smaller than the distance the grasp is from the next box in the workspace map. (See figure 6.10.)

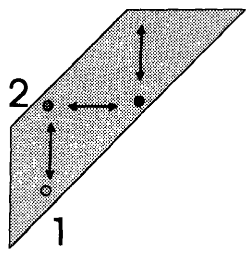

Figure 6.9 Figure showing that if a grasp is a distance of at least w from a forward edge, all following grasps will also be at least *w* from the forward edge. If grasp 1 is at least *w* from its forward edge, then taking a step of at least *w* in that direction ensures that at least a distance equal to the step size lies between the new grasp and the new forward edge. Since the smallest possible step size is *w*, once a step is taken, all following grasps will be at least a distance of *w* from the next forward edge.

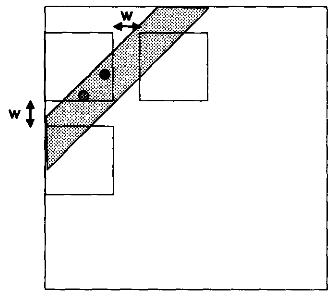

Figure 6.10 Band is of adequate size and is correctly placed to allow regrasps. However, the grasp shown by the lighter dot cannot be replaced by a new grasp in the forward direction until the object rotates sufficiently, bringing it to the position of the darker dot.

Therefore, rotating the object until a finger reaches the end of a workspace is sufficient to ensure that a new grasp exists, and it is also the most efficient, since it achieves the maximum rotation per regrasp.

If the 3 conditions are met, the following rules will successfully reorient the object counterclockwise. (The rules to rotate the object clockwise are in parentheses.)

1) Rotate counterclockwise (clockwise) until the grasp reaches the edge of its finger pair box, or until the distance from the grasp to the next finger pair box in the workspace map is less than or equal to the distance from the grasp to the upcoming edge in map space.

2) If the grasp is in an outer box, take a positive horizontal (negative vertical) step to the left (top) edge of the inner box.

3) If the grasp is an inner box, take a positive vertical (negative horizontal) step to the bottom (right) edge of the outer box.

4) Return to step 1.

Grasps gaits following these rules create a staircase-like pattern in the grasp map, as they are always taking alternate positive (negative) horizontal and vertical steps in the grasp map. (See figure 6.10 (a).)

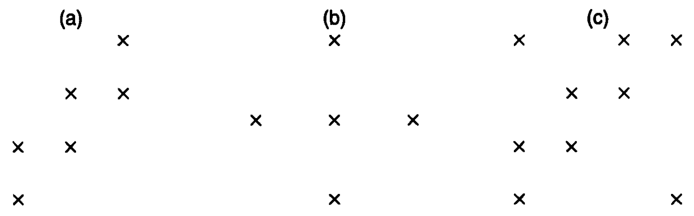

Figure 6.10 Typical gait patterns as seen from the grasp map: (a) staircase, (b) plus (+), (c) combined.

6.3.2 The Back-And-Forth Gait

More complicated than the move-forward gait is the back-and-forth gait. This gait is necessary when the grasp map is not continuous due to the object's eccentricity and forward moves in the grasp map cannot always be found. We approximate these grasps maps as a box and then ask the question: how big the box has to be to ensure a gait which will continually rotate the object? We have seen from figures 4.7 and 4.8 that for a discontinuous map, both forward and backward moves will be necessary for continual object rotation. We can state the following conditions regarding each type of move, similar to the conditions for the forward move within the band prototype.

1) In order to be able to make a forward move, the box needs to be at least w wide and tall and the grasp needs to be at least *w* from the forward edge. (See figure 6.11(a).) The box also has to pass through both workspace box pairs (inner and outer). (See figure 6.11(b).)

2) In order to make a move backwards, the box needs to be at least *w* wide and tall and the grasp needs to be at least *w* from the back edge. In addition, the object can only be rotated until one of the back edges of the box reaches a workspace edge. (See figure 6.11 (c).)

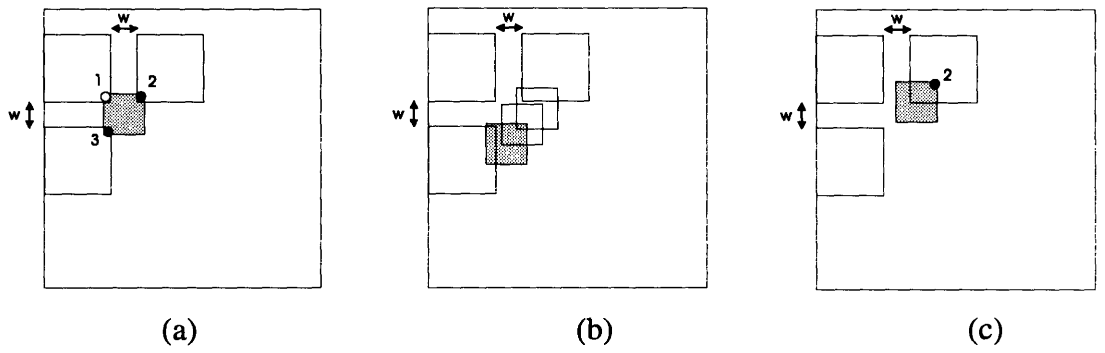

Figure 6.11 Figures showing conditions 1-3 for the box prototype. Figure 6.11 (a) shows that from grasp 1, a forward move to grasp 2 or a backward move to grasp 3 can only be made if the dimensions of the box are at least *w* x *w*. Figure 6.11 (b) shows that the box must intersect both workspace pair boxes if any grasp changes are to be made. Figure 6.11 (c) demonstrates that if the object is rotated too far a backward step cannot be made. Specifically, assuming grasp 2 shown in figure 6.11 (a), the object cannot be rotated past the orientation represented in the figure, since the left edge of the grasp map box would then leave the previous workspace box.

If the box is a square equal to *w*, eventually no rotation will be allowed for the following reasons. If the grasp is on a back edge, it will eventually hit a workspace edge, and once a forward move is made, it will not be able to rotate further, since no further jump ahead will be possible as condition 1 is violated, and no jump back will be possible, because the box will have left the previous workspace box. Therefore, if the grasp was on a front edge to begin with, it would not have been able to rotate further. If the box is bigger than omega, as you rotate the object, no matter what kind of moves are made (forward, rotate, backwards, rotate...), eventually the back edge of the box will reach a workspace edge. In order to rotate further, the grasp must be in the new workspace box (either by jumping there or being there already) which is at least *w* from the back edge and it must be able to make another jump forward, since condition 2 will be violated if rotation continues. In order to be able to make another jump forward, the grasp must be at least w from the front edge. Therefore the minimum box size must be 2w by 2w. If these conditions are met, the following rules will generate a gait for counterclockwise rotations. (Clockwise rotations can be generated by substituting right for left, top for bottom, positive for negative and visa versa.)

1) If the grasp is at least a distance w from both the top and right edge of the grasp map box, rotate until the grasp reaches the edge of its finger pair box. (We refer to this distance as the rotation limit.)

2) If the grasp is less than a distance w from the top of the grasp map box, calculate the distance that the object can be rotated before the bottom of the grasp map box will be a distance w from the bottom of the present finger pair box. (We refer to this distance as the upper limit.) Then rotate either the rotation limit or the upper limit, whichever is less.

3) If the grasp is less than a distance w from the right of the grasp map box, calculate the distance that the object can be rotated before the left of the grasp map box will be a distance w from the left of the present finger pair box. (We refer to this distance as the right limit.) Then rotate either the rotation limit or the right limit, whichever is less.

4) If the grasp is less than a distance w from both the right and top of the grasp map box, rotate either the rotation limit, the upper limit, or the right limit, whichever is least of the three.

5) If the object was rotated by the rotation limit, take either a positive horizontal or vertical step of size w, whichever is possible. Return to step 1.

6) If the grasp was rotated by the upper limit, take either a negative horizontal or vertical step, whichever is possible. Take the largest step possible.

7) If step 6 was just performed using a horizontal step, take the largest possible negative vertical step.

8) If step 6 or 7 was just performed, take a positive vertical step equal to *w*. Return to step 1.

9) If the grasp was rotated by the right limit, take either a negative vertical or horizontal step, whichever is possible. Take the largest step possible.

10) If step 9 was just performed using a vertical step, take the largest possible negative horizontal step.

11) Take a positive horizontal step equal to *w*. Return to step 1.

Using this method, if the conditions for the box size and placement are met, there is no initial condition that would preclude a successful gait. However, if, for a counterclockwise (clockwise) rotation, the grasp was less than *w* from the top or right (bottom or left) edge, and the opposing edge was in the same finger pair box, the object would first have to rotate backwards to get the grasp in a position that it could rotate continually in the desired direction. For a grasp map in which only a box of the required size fits, the above rules generate a plus (+) pattern in the grasp map, requiring exactly five points in the grasp map to rotate the object continually. For maps which include larger boxes, these rules generate a hybrid of + and staircase patterns. (See figure 6.10 (b) and (c).)

6.4 Pruning the Search Tree Using Prototype Gaits

As the smoothness of an object decreases, the grasp map becomes less regular but still follows the general trend for its given eccentricity. Therefore some grasp maps may not contain a band or box of sufficient size, but the missing pieces of the map may not be required for a gait to exist for that object. In that case, the object could be rotated using the same general gait pattern (either staircase or +), but the step sizes would need to be modified to fit the specific grasp map. For these cases a modified search is used which looks only for the general structure in the grasp map (either staircase or +) already known to work for that type of object, as opposed to the general search, which looks at any viable finger motion or rotation. For example, consider the grasp map for this pentagon. While a continuous band wont fit within the grasp map, a staircase pattern still fits within the map since the stable grasp regions lie in the same direction as the workspace pair boxes. In fact we have seen the staircase gait work in figure 4.6. As explored in the next section, this modified search using is much more efficient than the general search method at finding a path, and in general it is equally as efficient as the rule-based method.

6.5 Comparing Methods

This section compares results of the different approaches to generating gaits investigated in this thesis. The first approach is the general search, made feasible by using a best first technique and by not considering multiple paths through a node. These results were already discussed in section 5.4. The second approach is either the rule-based method if the sufficient conditions for such a method are met, or the modified search based on the rule-based method. The objects and initial conditions used for the best first search in section 5.4 were used again to compare methods for achieving a goal of reorienting the object 360°. (The rule-based methods will rotate an object continually, but a finite goal is needed for comparison.) Figure 5.3 shows the objects used and table 6.1 compares the results from the two methods using a coefficient of friction of .7. The three types of data used for comparisons are (1) the number of nodes opened, (2) the path length, and (3) the number of regrasps required.

For the modified rule-based method, the path length always equals the number of regrasps, since object rotations are implicit in the structure of the gait in the grasp map. For the rule-based search, all three measures are always equal, since no extra nodes ever need to be opened. Interesting comparisons between the two groups are (1) nodes opened - path length (showing an efficiency of the method) and (2) number of regrasps (showing an efficiency of the path found). Objects (a), (b) and (d) were successfully rotated using the rule-based methods, and object (c) required a modified search. Object (c) is a typical case in that, even though a modified search was used, its efficiency was 100% (the number of nodes equaled the path length). For the many examples we tried which used the modified search, the number of nodes opened tended to be equal or close to equal to the path length. These results show that the rule-based or modified rule-based method is much more efficient at finding a path than the best-first search. In addition, the number of regrasps required tend to be less for the rule-based approach.

DATA TYPE	GENERAL	RULE-BASED
	circle (a)	
nodes opened	957.7	9.0
path length	82.0	9.0
regrasps	41.1	9.0
	ellipse (b)	
nodes opened	14,486.3	13.8
path length	30.4	13.8
regrasps	20.3	13.8
	pentagon (c)	
nodes opened	208.2	14.5
path length	29.5	14.5
regrasps	18.0	14.5
	hexagon (d)	
nodes opened	1,421.2	18.6
path length	83.4	18.6
regrasps	58.0	18.6

Table 6.1 Table comparing results of general (best-first) search with the prototype-gait approach for a coefficient of friction equal to .7.

Chapter 7

Implementation

7.1 Overview

So far in this thesis methods for generating grasp gaits have been developed to manipulate convex objects in the plane using a robot hand. In this chapter the implementation of these grasp gaits using an actual robot hand is presented. Section 7.2 describes the hardware used for the robot hand and the experimental setup. Section 7.3 presents an overview of the control algorithms that achieve the desired finger motions and forces, as well as control the object motion. Section 7.4 presents a detailed description of each of the components in the controller. In section 7.5, the results of the implementation are discussed.

7.2 Hardware

The methods in this thesis generate gaits for a three-fingered robot hand. For each finger we use a PHANToM, designed and built by Salisbury and Massie. The phantom was designed to be used as a force-reflecting device for haptic interaction with virtual objects, but its good force control properties, which make it well suited for its intended purpose, also make it well suited for use as a robot. The phantom, shown in figure 7.1, is a three degree of freedom device connecting 3 DC brushed motors with encoders to its endpoint through a lightweight, aluminum linkage. The phantom is cable driven and was designed to have low mass, low friction, low backlash, high stiffness and good backdrivability. Its workspace is approximately a sphere with a diameter of 10 inches. The maximum force that the it can apply is about 2 pounds. The only sensors that the phantom uses are the encoders which measure the position of the motors.

Using three independent phantoms as a hand, there are 5 different coordinate frames involved in the implementation. The first is the common frame *C*, and can be thought of as the frame that would be attached to the palm of the hand. This frame remains fixed throughout the object repositioning, and the desired and actual object positions are always measured in this frame. The next coordinate frame is the object frame *O*, which is fixed to the object and whose origin is located at the desired center of rotation of the object. This is the same frame as frame *f*, used to discretize the object when generating the grasp map (from chapter 3). The transformation between the object frame and the common frame changes and can be determined from the object angle and the object displacement, measured in the common frame. The final three frames are each of the phantom frames. The phantoms are attached to a table such that when each phantom is in its home position the fingertips are approximately 120 degrees apart at a radius of about 4 inches as measured from the common frame. (See figure 7.1.) At these positions a phantoms' workspace is approximately 85 degrees for objects of an average radius of 3-6 inches.

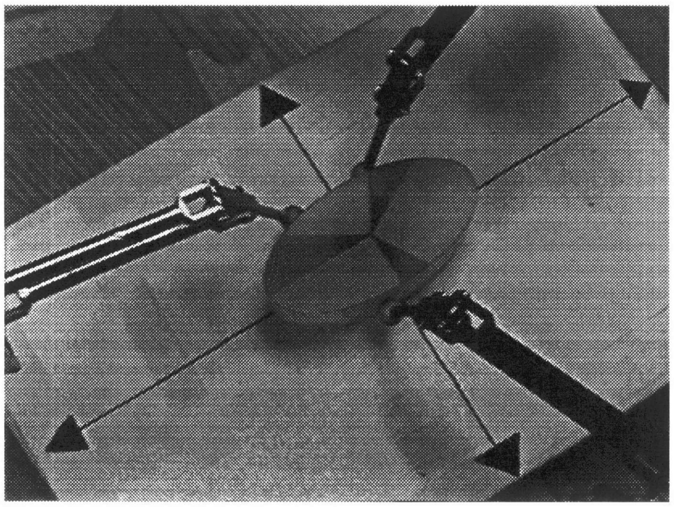

Figure 7.1 Phantom set up.

To calibrate the phantoms, they are put in their home position at the start of the calibration program, and the value of the encoders at that position is taken as the zero position. For use as a hand, each phantom is "shown" the position and orientation of the common frame by moving its endpoint to the origin of the common frame as well as to another point along its x axis. The z axis of the common frame is taken to be the same as for each phantom frame, which is in the upward vertical direction. Therefore the plane of rotation is the horizontal x,y plane. The transformation between each frame and the common frame remains fixed throughout the setup. Transformations between phantom frames are not done, but phantom interactions are done instead by transforming both into the common frame.

The fingers are assumed to have the ability to apply a force in an arbitrary direction through the fingertips, which are modeled as point contacts with friction. We include a small ball at the end of the tip to increase its effective workspace. By doing so, the phantom can contact a point on the object that is slightly behind the end point of its inner link. If the ball was not included, the lower link would have to intersect with the object in order for the endpoint to make contact at the desired location. However, the inclusion of the ball does introduce some rolling between the finger and the object. The extent to which this rolling is significant can be estimated by approximating the object to be of a fixed radius equal to the radius at the contact and the change in orientation of the finger as it moves to be zero. Then the equation relating the change in angles of the object and the finger position is given by

$$\theta_o = (1 + (r_f / r_o))\theta_f,$$

where θo is equal to the object angle, θf is equal to the finger angle, *rf* is equal to the finger radius, and *ro* is equal to the object radius. The radius of the fingertip ball is equal to .307 inches and an average object radius is 3 inches. Therefore the extra motion caused by using a ball instead of a point is approximately 1/10 of the total motion. This fraction is low enough that the balls can be modeled as points and rolling can be neglected. Further, any cumulative errors due to rolling will be handled by the object localizer/controller described later.

The objects used were generated on the screen with a mouse while using the gait planner, described in section 7.4, and then cut out of styrofoam. At the beginning of each gait the object was placed approximately at the center of the common frame with the common and object axes aligned. Exact placement of the object is not necessary, as the object is disturbed from its initial position during the initial grasping of the object. Therefore these and other errors must be handled and are done so by the object localizer/controller, described in the following sections.

7.3 Controller Overview

In this section the control algorithms used to implement a grasp gait are discussed. The major functions of the controller and the interactions between its major components are discussed. In the next section, a more detailed discussion of some of the subsytems is presented.

Conceptually, the system can be represented as shown in figure 7.2. An object geometry and a desired object motion is fed to a planner, which generates a nominal gait plan for the fingers and object. The controller has three main processes which serve to carry out the desired motion. The first process moves the fingers in a way that is expected to cause the desired object motion using PD control. The second process maintains the grasp, ensuring a nominal grip force is always maintained and resisting disturbance forces which would cause the object to slip by squeezing more tightly. This is achieved by feeding forward a nominal grip force, which is added to the calculated servo forces, while adjusting the magnitudes of the grip forces as described below. (The grip force portion of a finger's total commanded force is the force that is in the direction of the opposing finger involved in the grasp and is equal and opposite to the opposing finger's force. It is a homogenous solution to the equilibrium equation and therefore imposes no net force on the object.) The third process estimates the actual object position and changes the nominal plan to correct for any errors. The only observed output is the position of the fingertips.

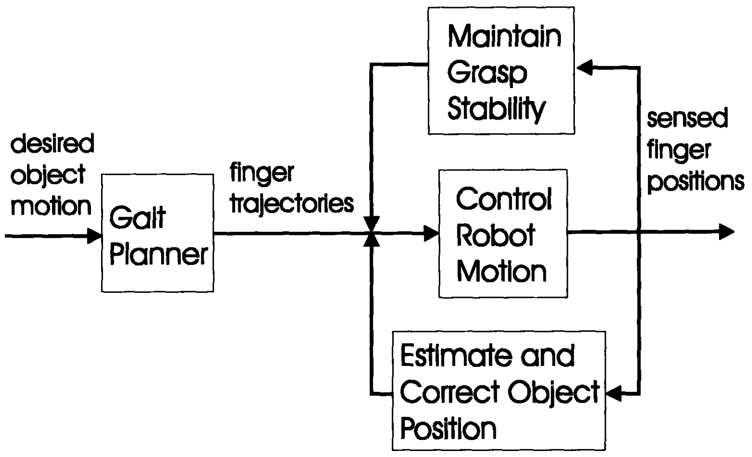

Figure 7.2 Conceptual System Diagram

It is interesting to note that with only position sensors we are able to control all three processes: finger positioning, grasp force maintenance and object tracking and correction. Controlling finger positions using position sensory information as feedback to a PD controller is a standard technique. However, controlling grasp forces using only position sensory information is more surprising. This is achieved by looking at the commanded servo finger forces coming out of the PD controller and ensuring that they always have a component in the desired grip force direction equal or greater in magnitude than the desired nominal grip force and that the forces are within the friction cone of the particular contact. If either of these conditions are violated, the specified grip force, which is fed forward and added to the commanded servo finger force, is increased, causing the fingers to squeeze more tightly. Using position sensors to control grasp force maintenance in this manner can be achieved because the phantoms are accurate force transducers.

Ensuring that the total commanded force always has a component in the grip force direction avoids the problem of having a finger break contact with the object. Since the fingers are under PD control, if the object is displaced far enough from its assumed position, either by a disturbance force or because the grasp on the object or the object itself is not where it is assumed to be, even though a nominal grip force is added to the servo force, the net force on one or both of the fingers will eventually reach zero. Consider the case where there is a disturbance force in the direction of the line between the two fingers grasping the object. The virtual springs in the servo act as a restoring force and pulls or pushes the fingers back to their intended position. The grip force of one of the fingers acts in the same direction as the spring, while the grip force of the other acts as a force pulling away from the desired position. Eventually these two opposing forces will reach an equilibrium, and the total commanded force of one of the fingers will equal zero. If the object is displaced far enough from its intended position, without any sort of correction, that finger will break contact. Figure 7.3 illustrates this condition.

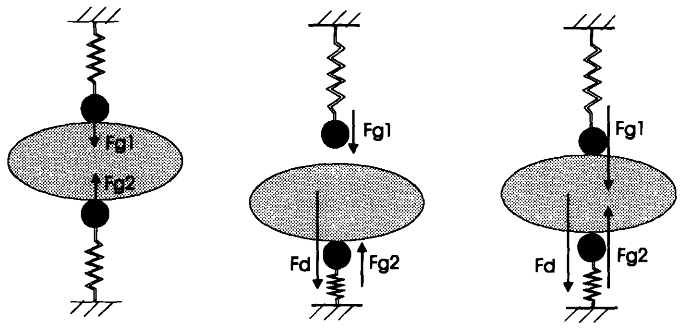

Figure 7.3 Series illustrating that a finger may break contact with the object in the presence of a disturbance force if only a nominal grip force is used. The springs represent the commanded servo forces of the fingers and the arrows labeled *Fg* represent grip forces. If grip forces are increased so that the total commanded forces always have a positive grip force component, then contact will not be broken.

Ensuring that the total commanded force always lies within the friction cone of the contact avoids the problem of slipping. Consider the case of a disturbance force on the object acting perpendicular to the line between the two fingers grasping the object. Since there is some grip force between the two fingers, as the disturbance force pulls on the object, there will be a force transmitted to the fingers in the same direction as the disturbance force, due to the fact that friction is keeping the object from slipping. This force on each of the fingers will tend to displace the fingers in the direction of the force. Since the fingers are servoing to some initial position, the position controller will respond by commanding the fingers to apply a force in the direction opposite the disturbance force. If the disturbance force increases, eventually the total commanded force of the fingers, equal to the sum of the grip force and the servo force, will no longer lie within the friction cones at the contact points, and the fingers will slip relative to the object as they maintain their commanded position while the object is pulled away. If the grip force is increased whenever the total commanded force on a finger lies outside of its friction cone, slipping can be avoided. Figure 6.4 illustrates this condition. However, there is no way to distinguish between a disturbance force which acts on the object and is transmitted to the finger through friction from a disturbance force which acts directly on the finger. Therefore, if a disturbance force acts directly on the finger, the grip will also tighten even though there is no real danger of slip occurring.

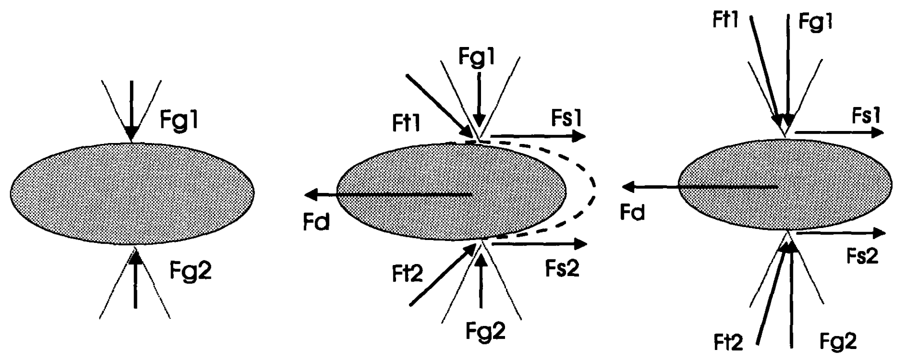

Figure 7.4 Series illustrating the need to increase grip forces in the presence of a disturbance force in order to keep the total applied force within the friction cone. *Fgi* represents a finger's grip force, *Fsi,* a finger's servo force, and *Fti,* a finger's total force. *Fd* represents a disturbance force acting on the object.

The third controller process estimates the current object position using finger position sensor information and modifies the planned gait or trajectories so that the actual object trajectory is brought back in line with the desired one. A more thorough discussion of how this is accomplished is given later in this section and in the following one. Here we will just state that this estimator succeeds without normal information because it relies on the previous object position estimate and searches a small local space around this for the most likely new object position.

A fourth task is determining when a finger has made contact with the object. This is achieved by setting a threshold force to be some amount greater than the forces generally required to move a finger along a trajectory when it is not in contact with the object. Then when a finger is moving along a trajectory that is intended to bring it in contact with the object, once the commanded servo forces exceed the threshold force for a specified number of servo cycles, the finger is assumed to have made contact with the object. This method requires, of course, that the commanded trajectory extend beyond the surface of the object, so that the commanded servo forces increase once contact has been made, since motion cannot continue although the desired position continues to move. Observed commanded servo forces tended to be fairly consistent at around .5 ounces, so the threshold force was set to 1 ounce. Exceeding the threshold force for 3 cycles was required before contact was assumed.

A top level block diagram representing the basic components of the controller is shown in figure 7.5. The input to the overall system is an object geometry, initial finger positions, and a desired final object position. This information is fed into a gait planner, whose output is intermediate finger positions and object positions, or intermediate grasps and rotations. These outputs are the inputs to the trajectory planner. The trajectory planner outputs detailed finger position trajectories, the magnitudes of the internal forces between each pair of fingers, and contact normals if a finger is in contact with the object. The detailed finger position trajectories are fed through a PD controller. The output of the PD controller is a set of forces. These forces are added to the internal grip forces described below. The sum of these forces are converted to currents which are fed to the phantom motors. The result is a motion or force at the phantom finger tips. These motions or forces interact through the object and cause some object motion. The only measured outputs are the positions of the fingertips. The measured fingertip positions are fed back to the PD controller, to the trajectory planner, and to the following other subsytems. The measured positions, along with the internal force magnitudes are fed to the internal force generator, whose output is a set of forces which are the internal forces for each finger. These forces are summed with the servo forces and fed to the phantoms. The measured fingertip positions are also fed to the object localizer, whose output is the estimated intermediate position of the object. This estimate is fed back to the trajectory generator. The measured fingertip positions are also fed to the grip force and friction cone check, along with the total commanded forces. The output of this check is adjustment magnitudes for the internal grasp forces. These magnitudes are fed back to the internal force generator.

There are 3 different servo rates occurring throughout this system. The fastest is the inner servo loop of the fingers, running at approximately 3500 Hz. This loop includes the PD controller, the phantoms, the object, the internal force generator and the grip and friction cone check. The next is the middle loop which runs on the order of 1 Hz. This loop includes the trajectory planner and the object localizer. The final rate is the time it takes to generate and complete a full reorientation which is on the order of one minute.

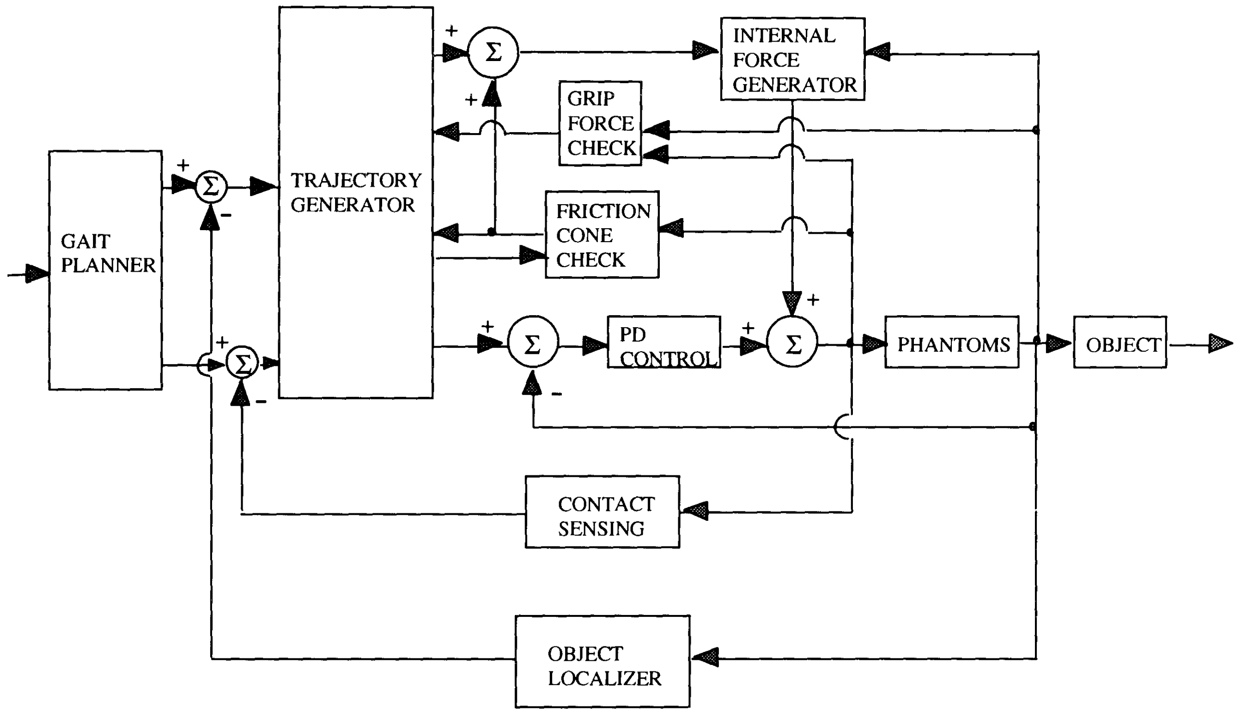

Figure 7.5 Overview Block Diagram

7.4 Subsystem Descriptions

7.4.1 The Gait Planner

Inputs and Outputs:

Next we discuss the inputs and outputs to the gait planner subsytem. The first input to the gait planner is an object geometry. This information is of the following form: the number of vertexes of the object, followed by the x,y coordinates of each vertex pair, either read from a file or read as input from the screen where the mouse is clicked. If the object is an ellipse, a value of 0 is given as the number of vertices, followed by the size of the major and minor axis. The second input to the gait planner is an initial two fingered grasp. This is either read from a file as two finger angles, or is read from the screen where the mouse is clicked on the grasp map (since one point on the grasp map represents a two-fingered grasp). The final input is a desired final object orientation.

The output of the gait planner is a sequence of intermediate finger positions and object rotations along with a letter signaling which type of data it is written to *motion_file*. It is given in the following form:

```
i  90  -150 270
r   30
t   -90  126 270  ...
```

The first entry in each line is either an i, r, or t, standing for 'initial grasp', 'rotation' or 'transition'. The initial grasp is self-explanitory. The rotation is an object rotation, and the transition is a transition between grasps. If the letter is an i or t, it is followed by three numbers, which are the positions of fingers 1,2 and 3 respectively, given as angles measured relative to the object coordinate frame. The two fingers that are in contact with the object are given as positive values. The radius of the finger positions are either determined by the object geometry if they are in contact, or is equal to some value *max_r* if the finger is not in contact. The value of *max_r* is chosen to be large enough so that as the finger moves at a constant radius to make contact with the object at another angle it is sure not to collide with the object. For most objects, *max_r* was set to be equal to 1.5 times the maximum object radius. If the letter is an r, it is followed by one number, which is the intermediate change in object orientation (delta theta) given in degrees as measured from the common frame. Each line of *motion_file* is an input to the trajectory planner, which is the next subsystem discussed.

7.4.2 The Trajectory Planner

Inputs and Outputs:

The trajectory planner takes as input either an initial grasp, a rotation, or a transition, along with the corresponding data described above. It also takes as input the last desired position of each finger commanded by the servo loop. The reason for this is that a new trajectory must begin with the last desired position from the previous trajectory. If during the last trajectory, a finger made contact with the object, the last desired position is not known in the trajectory planner, since the trajectory extends into the object and is stopped when the threshold force is reached. The output of the trajectory planner is a two dimensional array for each finger. The size of each array is n x 8, where n is equal to the length of the trajectory which is determined as described below. For each step in the trajectory, the first three columns of the array are the x, y, and z positions of the fingertip in phantom coordinates for that particular finger. The second three columns are the magnitudes of the internal forces between the present finger and fingers 0,1 and 2, respectively. At least one of these values will always be zero, since the internal force between a finger and itself will always be zero. The last two columns are the x and y components of the inward unit normal of the object at each contact, in object coordinates. If the present finger is not in contact with the object, these values are both zero. The trajectory planner also outputs the type of each trajectory. The options are 0 for a finger that will be applying no forces and will not be contacting the object, 1 for a finger that will be applying transition forces, 2 for a finger that will be making contact with the object, and 3 for a finger that will be applying forces that are not transition forces. This information will be important for the contact sensing and grip and friction cone checking subsystems.

Determining the path:

For each input, the trajectory planner must first determine the general path of each finger. The path always begins with the last desired position commanded by the servo loop. If the input is a rotation, the general path of each finger in contact with the object is to travel in an arc at the present radius and increasing by the angle given as input as measured in the common frame. If there is a finger that is not in contact with the object, its desired trajectory remains constant. If the input is a transition, several trajectories follow. First, the 'new' finger (the one which had a negative value at the last transition) is brought to the correct angle by following an arc at the constant radius of *max_r*, and is then brought to the correct position on the object by following a line at the present angle inwards toward the object until contact is made. Then the three fingers servo to the same position while the internal forces change as described below. Finally, the third finger which is not involved in the new two-fingered grasp is then removed from the object by following a line outward at the present angle until the radius equals *max_r*.

This third finger can be in general used as a stabilization finger, by bringing it in contact with the object during rotations, even though it is not actively applying a grip force. This is especially helpful during 3-d rotations, when the object may tend to roll out of the plane due to changes in fingertip orientation which cannot be controlled. Adding a third finger helps alleviate this problem. In 2-d, a stabilization finger is not as necessary for stability but is extremely helpful for the object localization algorithm. Therefore the third finger is included to aid with stability, to increase the effectiveness of the object localization algorithm and so that the code can extend to planar rotations in 3-d. With this finger included, the transition begins by removing the third finger by having it follow a line outward at the present angle until the radius equals *max_r*. The transition ends by adding the new third finger (the one that was removed in the transition) in the same fashion that the new finger was added. The angle which it is moved to in the common frame is equal to the midpoint of its workspace minus half of the maximum rotation. This ensures that its trajectory during rotation will never be outside of its workspace. (see pictures) The transition trajectories as described in the previous paragraph occur between the above initial and final trajectory if the third finger is contacting the object during rotations.

To calculate the specific cartesian coordinates of the trajectories, the line or arc over which the finger is to travel is broken up into *len_tra* peices, such that the distance between each set of coordinates is of length *res*, which is an input of the trajectory planner. An example value of *res* is 1/10 inch. These values are then transformed into the respective phantom coordinates and passed to the PD controller.

Determining the internal force magnitudes:
To determine which fingers apply internal forces with which other fingers, the type of step is considered. If the input is a rotation, then the only internal force pairs that is non-zero is the pair of fingers involved in the two-fingered grasp, which are the two fingers that have non-negative entries in the last transition step. The magnitude of the internal force between these two fingers is constant throughout the array and equal to the sum of the variable *grip_force*, which is an input to the trajectory planner, and the adjusted grip force magnitude which is fed back to the trajectory planner from the servo loop. In other words, if during the last trajectory the internal force between those two fingers had to be increased in order to resist a disturbance force, the amount of the increase is fed back to the trajectory planner so that when the next step in the gait occurs, there is a continuous grip force that is applied to the object.

If the present gait step is a transition, during trajectories where a finger is contacting or being removed from the object, internal forces between the two gripping fingers are calculated to be constant in the same way as for rotations. During the trajectory where there are three fingers on the object servoing to a constant position, the internal grip force is transferred from the old pair of gripping fingers to the new pair. The grip force increases linearly from 0 to *grip_force* between the new pair, and decreases linearly from the last implemented grip force to 0 between the new pair.

Determining the unit normal components:
The last two columns in the trajectory array are the x and y components of the inward unit normal at the contact point measured from the common frame. These are determined from the object geometry in the object frame and then converted to the common frame. If the finger is not in contact with the object during a particular trajectory, both values are zero.

7.4.3 The Internal Force Generator

The internal force generator takes as input the measured finger positions, the grip force magnitudes from the trajectory planner, and the adjusted grip force magnitudes from the grip and friction force check and calculates as output the commanded internal forces. These forces are added to the commanded servo forces and sent to the phantoms. Internal forces are bias forces which do not disturb the equilibrium of the object, since each is equal and opposite to the force applied by the opposing finger. The direction of each grip force *Fgi* is given by $Fgi/\|Fgi\| = (Rj-Ri)/\|Rj-Ri\|$, where *Rj* and *Ri* are the measured positions of the fingers *j* and *i*.

7.4.4 The Grip Force and Friction Force Check

The motivation for the grip force and friction cone check was given in section 7.2, so here only a mathematical description will be given. With respect to the grip force check, the total commanded force Fit is transformed into common frame coordinates. If for a particular finger one of the grip force magnitudes is non-zero, the dot product between the total force and the grip force direction for that finger and grip force magnitude is taken. If the value is less than the nominal grip force, a fixed amount is added to its grip force value, as well to the opposing finger's value. This fixed amount is equal to the quotient of the nominal grip force and *settle,* where *settle* is the number of servo cycles specified to apply a nominal grip force and is an input into the program. It was set equal to 40 for most runs. The reason to add only a fixed amount to the grip force each servo cycle is so that a continuous force is always applied by each finger.

With respect to the friction cone check, the total commanded force is normalized and then transformed to the common coordinate frame. Its dot product is taken with the unit normal for that contact, and the interior angle is calculated. If the angle is greater or equal to the friction cone angle, which is an input into the program, the appropriate grip forces are adjusted appropriately (as described above). However, since the exact position of the object is not known, there is a possibility that the estimated friction cone is incorrect, and a force which in fact lies within the actual friction cone is calculated to lie outside of it. To avoid instabilities, the actual grip force is also checked to ensure that it lies within the estimated friction cone. If it does not, no extra force is added. (This can occur if the object localizer changes its estimate of position after a finger has already been added to the object.) Further, a saturation of 1 lb. is included, so that if the total force is above this value, no additional force is added.

7.4.5 The Object Localizer

The object localizer takes as input the expected object position, the measured positions of the fingers, and a model of the object shape. Since the actual object position and/or the grasp relative to the object may have migrated from their expected positions, the actual positions must be estimated in order that the next trajectories be modified to bring the object and grasps back to the original gait plan.

To estimate the object position, four reference frames must be considered. The first is the fixed reference frame, *F*, from which the positions of the next three frames are measures. The other three frames are the expected object frame *E*, the trial object frame *T*, and the actual object frame *A*. The expected object position (xe, ye, and θe as measured from frame *F*) is used as a starting point. From there, a local space around this point is searched to find the position that minimizes finger position errors in the following way. When the object localizer is used, all three fingers are known to be in contact with the object. These finger positions are converted to polar coordinates in the trial object frame. The trial object frame in relation to the fixed frame is given by xt, yt, and θt, where these values are one of the points in this space near the point representing the expected object frame. Then, given each finger angle, a finger position error is calculated which is the square of the difference between its actual radius in the object frame and its expected radius, which would be the object radius for that angle, since the finger is assumed to be in contact with the object. The total error, E, for a given trial object position is defined as

$$E = \sum_{i=1,3} (Rt - Ra)^2,$$

where Rt is the expected finger radius for a given trial object position, and Ra is the actual finger radius. The point (xt, yt, θt) in the local space neighboring the expected object position with the smallest error is found and is defined as ($xmin$, $ymin$, θmin).

Due to errors in finger position measurements, calibrations of the individual phantoms and transformations between coordinate frames, the trial position for the object frame with the minimum error is not necessarily the actual position of the object. Therefore some consideration has to be given to how much the new estimated position should move from the expected position to the position with the minimum calculated error. Specifically, certain directions within the space of possible positions may be more sensitive to changes in object position estimates than others. Therefore a weighting for each coordinate is needed such that

$$\begin{aligned} Xe_{new} &= W_x\, Xe_{old} + (1 - W_x)\, X_{\min}, \\ Ye_{new} &= W_y\, Ye_{old} + (1 - W_y)\, Y_{\min}, \\ \theta e_{new} &= W_\theta\, \theta e_{old} + (1 - W_\theta)\, \theta_{\min}, \end{aligned}$$

where Wx, Wy, and $W\theta$ are the weighting factors for each coordinate, and Xe, Ye, and θe are the estimated object positions. Consider the ellipse and the three fingered grasp shown in figure 3.1. If a trial position is considered which deviates from the actual position only in its y value, the error will be much larger than if a trial position deviates in its x value by the same amount. After trying a variety of different options, a weighting of

$$\begin{gathered} Wi = 1 / (1 + \partial^2 E / \partial i^2), \\ i = x, y, \theta, \end{gathered}$$

similar to the Levenberg-Marquardt approach for finding a local minimum. If the curvature is high, then moving in that direction will lead to a local minimum. Therefore the weight will be small so that coordinate of the new value will be favored. If the curvature is low, varying that coordinate does not provide much information or change in error. Using the above formulation, the weight will be high and the old value will be favored. These weights provide a kind of filtering of the new estimates, causing then to be less noisy but slightly lagged.

The output of the object localizer is a new object estimate. If the difference between the estimate and the desired position is larger than some specified allowable error, the next trajectory is one which attempts to correct that error directly, by moving the object to the correct location. If the error is less than the allowable error, the error is not immediately corrected, although the error is tracked, so that when a finger is added, the desired point of contact is adjusted to account for the deviation in object position. In addition, during a rotation, if there is an error in object angle, the amount rotated is adjusted to correct for this error.

7.4.6 The PD controller

The PD controller receives as input course position trajectories as well as measured finger positions. In the PD controller, a constant desired velocity is calculated and commanded as the phantom moves between these coursely specified points. A desired position is also commanded, calculated from the desired velocity and the measured time lapse between servo loops. The number of steps or servo loops taken between points given by the trajectory planner is specified by the variable *delay*, which is an input to the program. Average values of delay are equal to 200. Actual velocities are calculated from measured positions and filtered using a low pass filter with a coefficient of 1/10. The commanded servo force is then given as *Fsi = kp (x_d - x_m) + kv (v_d - v_m)*, where *Fsi* is the commanded servo force, *kp* is the proportional gain, *kv* is the derivitive gain, *x_d* is the desired position, *x_m* is the measured position, *v_d* is the desired velocity, and *v_m* is the measured velocity.

7.5 Discussion

Using the control algorithms described in the previous sections, we were successful in implementing a variety of gaits using the three-fingered robot system. A subset of the objects tested were those shown in figure 5.3, also used to compare gait generation methods. The gaits were used to rotate the circle, ellipse, pentagon and eccentric hexagon 360° in the plane. They were generated by the rule-based and modified rule-based methods.

In order to constrain all of the object forces to lie in the plane of rotation, the objects were rotated while sitting on a table. Grasp failure was able to occur, since grip forces were applied and the object could slip through the fingers if the grasp was not force-closure. The grip force and friction cone check were verified to work by tugging on the object while it was being grasped. When these checks were not on, the object slipped through the fingers of the object or a finger broke contact with the object. When these checks were on, the object did not slip and both fingers remained in contact with the object, regardless of the direction the object was displaced. The success of the object detection algorithm was also evident. During rotation and regrasping, small errors in object placement occurred, and these were seen to be corrected by the object localizer which operated between rotation and finger placement trajectories. If the errors were over 1/4 inch in any direction, the object would be recentered. Any orientation error was corrected during each rotation phase. Most importantly, the program successfully rotated the objects 360°. Even in cases where grasps were close to their friction cone and slip occurred, the robot system detected the change in position and recovered the gait.

Chapter 8

Conclusions

8.1 Summary

This thesis presents an analysis of grasp gaits for reorienting convex objects in the plane using a three-fingered robot hand. A method for generating gaits for any object shape was developed. In most cases, rule-based methods, which are efficient in both gait generation and implementation are sufficient for finding a solution to the problem of reorienting an object by any desired amount. In special cases, these rule-based methods are not sufficient and a best first search can be used to find a solution if one exists. Further, these gaits were demonstrated to work using a three-fingered robot system, using a few simple control algorithms and position sensor information.

8.2 Future Work

This thesis raises a number of interesting questions for further study. Two main areas will be discussed here. The first is how this work can be extended to manipulate objects in three dimensions. Assuming point contacts, a grasp of three fingers is required to constrain an object. Four fingers are then necessary to manipulate using grasp gaits. If only a planar slice of the three dimensional problem is considered (while forces are allowed to lie outside this plane) the grasp map becomes a three-dimensional space. The simplified grasp map prototypes and resulting gaits may extend to this domain. In fact, the two-dimensional grasp maps studied in this thesis are planes intersecting the volume of these three-dimensional grasp maps. If two fingers in the three-fingered grasp, three-dimensional map space are considered to be at the same point on the object, the map reduces back to the two-dimensional map, which is the plane equal to i=j. If the entire 3-d problem is considered, three-fingered grasps are required with 2 position variables, and the grasp map becomes a 6-dimensional space.

The second area of future work involves manipulating unknown objects. Since the rule-based methods require several orders of magnitude less searching than the best first search method, perhaps object shape and its associated grasp map can be estimated while looking for new grasps in regions of the object expected to be stable using the estimated grasp map and the corresponding. Modifying rule-based solutions should result in little searching and be efficient in generating solutions.

References

[1] Abell, T. and Erdmann, M. Stably Supported Rotations of a Planar Polygon with Two Frictionless Contacts. IEEE/RJS International Conference on Intelligent Robots and Systems, vol 3, pp. 411-418, Pittsburgh, PA, August 1995.

[2] Brock, D. Enhancing the Dexterity of a Robot Hand Using Controlled Slip. TR992, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May 1987.

[3] Chen, I-M and Burdick, J. W. A Qualitative Test for N-Finger Force-Closure Grasps on Planar Objects with Applications to Manipulation and Finger Gaits. IEEE International Conference on Robotics and Automation, pp. 814-820, Atlanta, GA, 1993.

[4] Fearing, R. Simplified Grasping and Manipulation with Dexterous Robot Hands. IEEE International Journal of Robotics and Automation, 2(4), 1986.

[5] Ferrari, C. and Canny, J. Planning Optimal Grasps, Proc of the 1992 IEEE International Conference on Roboics and Automation, Nice, France, May, 1992.

[6] Goldberg, K. Feeding and Sorting Algorithms For The Parallel-Jaw Gripper, Proc. 6th Int. Symposium on Robotics Research, 1993.

[7] Hanafusa, H. and Asada, H. Stable Prehension By A Robot Hand With Elastic Fingers, Robot Motion Planning and Control, M. Brady (ed.), pp. 323-336, MIT Press, 1982.

[8] Hong J. et. al. Fine Manipulation with Multifingered Hands. IEEE International Conference on Robotics and Automation, pp. 1568-1573, Cincinatti, OH, May 1990.

[9] Jacobsen, S.C. et. al. Design of the Utah/MIT Dexterous Hand. Proc. 1987 IEEE International Conference on Robotics and Automation, San Fransisco, April, 1987.

[10] Kerr, J. and Roth, B. Analysis of Multifingered Hands. International Journal of Robotics Research, 4(4):3-17, 1986.

[11] Lakshminarayana, K. Mechanics of Form Closure, ASME paper 78-DET-32, 1978.

[12] Li Z. and Sastry, S. Task-Oriented Opitmal Grasping by Multifingered Robot Hands, IEEE Jounal of Robotics and Automation, vol. 4, pp. 32-44, February, 1988.

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

[14] Montana, D. The Kinematics of Multi-Fingered Manipulation, IEEE Transactions on Robotics and Automation, vol 2, No. 4, pp. 491-503, August 1995.

[15] Nguyen, V-D. Constructing Force-Closure Grasps. International Journal of Robotics Research, 7(3):3-16, 1988.

[16] Okada, T. Object Handling System For Manual Industry, IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-9, no. 2, 1979.

[17] Omata, T. and Nagata, K. Planning Reorientation of an Object with a Multi-Fingered Hand, Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 5(4)4:3104-3110, 1994.

[18] Pollard, N. S. Parallel Methods for Synthesizing Whole-Hand Grasps from Generalized Prototypes. TR 1464, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1994.

[19] Rus, D. Coordinated Manipulation of Polygonal Objects, IEEE/RJS International Conference on Intelligent Robots and Systems, 1993.

[20] Salisbury, J. K. and Craig, J. J. Articulated Hands: Force Control and Kinematic Issues, Int. J. of Robotics Research, Vol. 1, No. 1, 1984.

5138-8