Fine Motion Planning
For Dexterous Manipulation

Daniela Rus
Ph.D Thesis

92-1323
December 1992

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

FINE MOTION PLANNING FOR DEXTEROUS MANIPULATION

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Daniela Rus
August 1992

FINE MOTION PLANNING FOR DEXTEROUS MANIPULATION

Daniela Rus, Ph.D.

Cornell University 1992

This thesis investigates the problem of dexterous manipulation; how can robots affect the world around them by means of their end-effectors? Dexterous manipulation is fundamental to robots operating intelligently and independently in their environments and it is a special motion planning problem.

Since the general motion planning problem with uncertainty is NEXP-hard, effort must be directed to defining classes of tasks that are tractable. We consider the reorientation problem: for a given robot hand, an arbitrary object, and a desired orientation with respect to the hand, find an algorithm to synthesize a robust plan for the fingers that accomplishes the desired reorientation. A reorientation algorithm devised for a robot hand should satisfy several properties. First, it must be able to accomplish arbitrarily large rotations. Second, since it must be implemented on a real device, it should involve simple finger motions that can be computed fast. Third, since the application domain is characterized by uncertainties that manifest themselves as imprecisions in calculations and inaccuracies in control, it must exhibit good stability properties. We propose algorithms for the reorientation problem that satisfy these properties. The basic idea is to use

some of the robot fingers to constrain the motion of the object and others to generate motion. This results in the idea of finger tracking as a high-level primitive for manipulation. We also propose an algebraic framework for manipulation that is theoretically well-founded and in which it is possible to use systematically and effectively the differential equations describing the interaction between objects in contact. Finally, we describe a simulator for the reorientation of polyhedra by finger tracking.

Table of Contents

List of Figures

Chapter 1

Introduction

This thesis investigates the problem of *dexterous manipulation* , which is the computational study of how robots can affect and learn about the world around them by means of their end-effectors. A long-standing dream of the robotics community has been to create robots that are capable of performing autonomously meaningful tasks, but progress has been slow. The dexterity of robots involves the study of a special class of motion-planning problems. Dexterous manipulation is fundamental to robots operating intelligently and independently in their environments, and results in this area contribute towards the computational study of autonomous behavior.

Automation has become an integral part of certain industrial activities, but one of the great deficiencies of today's robots is their lack of flexibility. Most industrial robots are capable only of simple and repetitive tasks, such as spray painting and spot welding. There are two main reasons for this deficiency. First, typical end-effectors use a very simple two stick configuration. Second, the use of mechanical hands with complex structure and independently moving fingers is not well understood. The design of *articulated hands* [MS85,JIK$^+$86] was inspired by the human hand, which is one of the most versatile known mechanisms for dexterous manipulation. The hand is capable of a wide range of tasks and it is important to understand how it works and what it can do, in the hope of duplicating some of its abilities for robots. Articulated hands are anthropomorphic mechanical hands with an opposing thumb and independently moving fingers. They are more versatile than conventional two-stick grippers. The larger number of joints allows them to adapt to different shapes, and thus, grasp well a larger class of objects. The larger number of fingers allow greater flexibility in constraining the number of degrees of freedom of a grasped object, to allow only certain types of motion. The extra fingers also allow flexibility in the repositioning of a grasped object within the grasp. They have high mechanical rigidity, and a wide spectrum of operating temperatures and pressures. Effective techniques for dexterous manipulation would have applications within a wide range of areas, including industrial assembly, and exploration of hazardous and remote environments.

Robots interact with their environment by sensing the environment's structure, analyzing it, and issuing actions to effect changes. In order for robots to operate intelligently and independently, they must be able to represent, reason about, and manipulate physical objects. To this end, high-level plans are needed to determine global actions. Methods are needed to select manipulation strategies to achieve the lower level goals as set by high level plans. For example, we would like to tell a robot to join two parts with a screw, and have it automatically synthesize the operations needed to accomplish the task.

1.1 An Example

Consider the the task of joining together two parts with a connecting part, a frequent occurrence on the assembly line. A simplified example of such a task is given in Figures 1.1 through Figure 1.4. Figure 1.1 shows three objects of simple geometry that are to be joined together. Part A is a parallelepiped with a triangular cavity. Part B is identical to Part A . Part C is a triangular parallelepiped that matches the holes of the other parts.

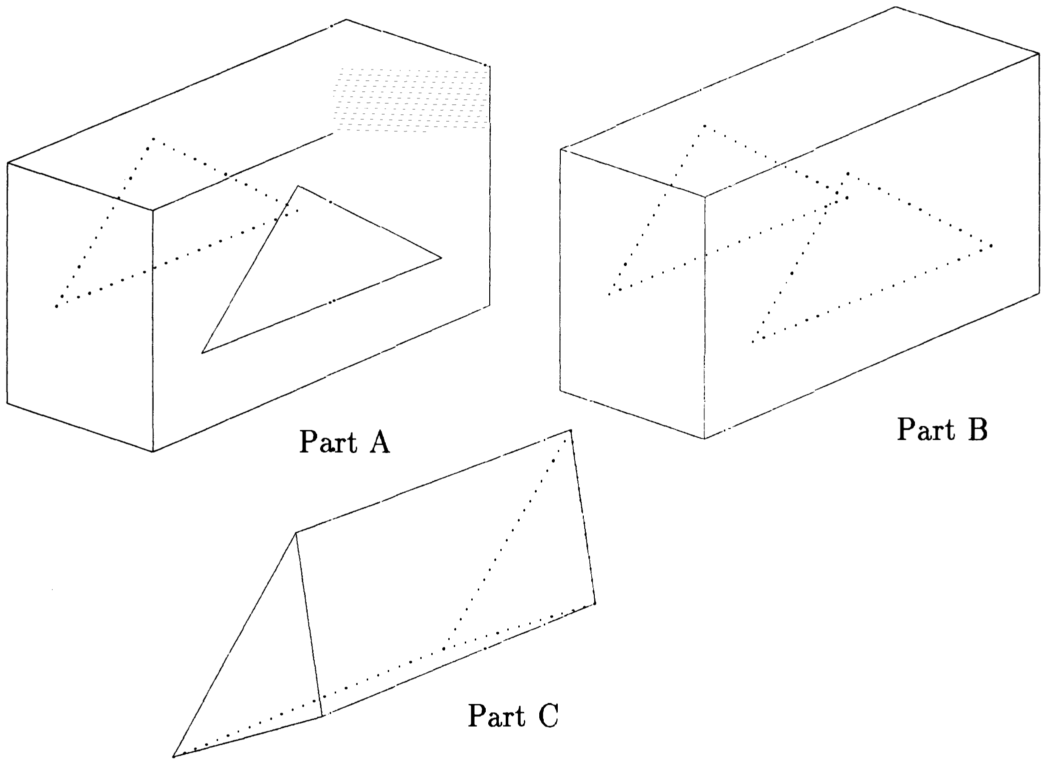

Figure 1.1: Three parts to be joined together

A robot executing the task of joining Part A to Part B with Part C must first pick up the joining part, Part C, and hold it securely. The orientation of the grasped part may not align with the profile of the hole. For easy insertion, the part should be reoriented, which necessitates adjustments within the grip. Figure 1.2 shows the new orientation for Part C after being adjusted to match the hole.

With the correct orientation for the joining part, the robot can insert Part C in the hole of Part A . Figure 1.3 shows the configuration of the assembly after this operation. In this example, the hole of Part A is polyhedral and the joining of the two parts is a peg insertion operation. Most applications of this nature involve combining two parts with a screw. When the hole of Part A is threaded and Part C is a special screw such as a hex-nut or a wing-nut, the connection is done by the more complicated operation of continuously rotating the screw inside the hole.

Part B can be added to the assembly in a similar fashion: it should be picked up, reoriented, and then merged with the rest of the assembly. Figure 1.4 show the final assembly.

The execution of our example task requires a robot capable of acquiring an object in its gripper and of performing actions with this object. The essential action that the robot must perform is to change the orientation of the grasped object without dropping it. Reorientations need to be performed to adjust the position of Part C so as to be aligned with the hole. Rotational motions are also employed in the insertion phase, when the hole is threaded. One way of performing reorientations is to use robot hands with revolving wrists. Although this simple solution is sometimes useful, it is far from being sufficient. The main reason is that the rigid attachment of the object inside the grip only gives rotations about one axis, but robots should be able to perform reorientations about different axes. Better solutions are those that exploit the multifinger structure of the gripper and use finger motions rather than large arm motions to achieve arbitrary rotations.

Most current implementations of assembly tasks similar to this example are based on special purpose programs. The robot programmer develops the code for the particular sequence of operations needed. The program is complicated further by the low-level control issues of the interaction between the robot and

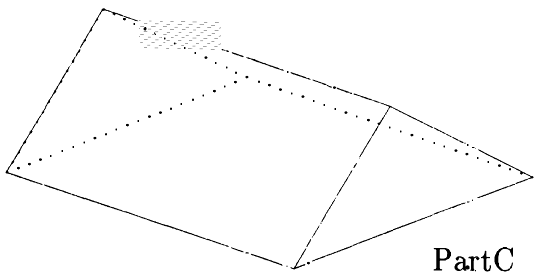

Figure 1.2: The reoriented connecting part

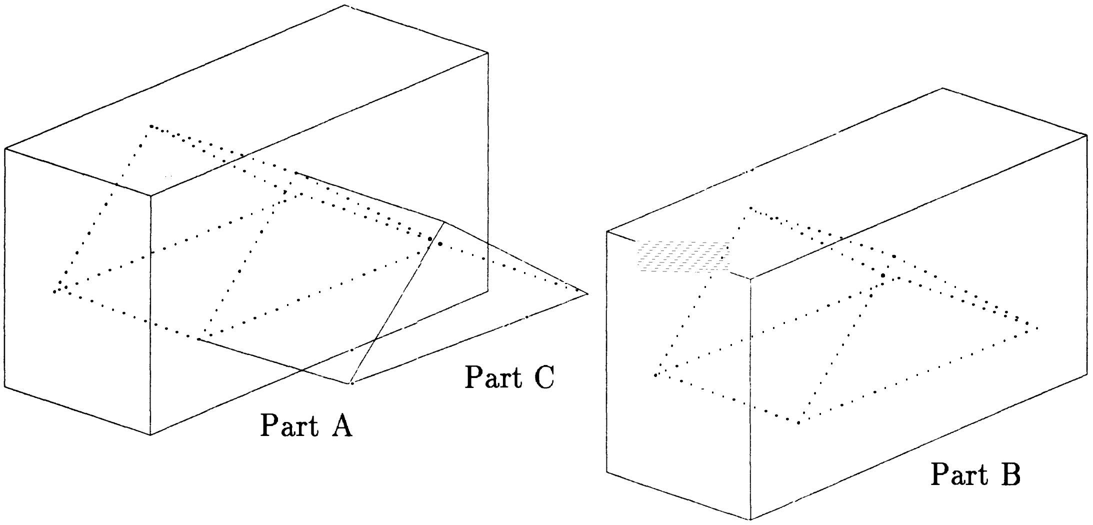

Figure 1.3: Part A joined with Part C

the object. A general approach, based on methods to synthesize the sequence of operations necessary for an arbitrary task and on strategies to use the low-level tools systematically is needed. The agent that determines the global strategies for a robot is a robot planner. For this example, a robot planner takes as input the task to join Part A to Part B with Part C, and automatically generates the commands to grasp Part C, reorient Part C, insert Part C onto Part A, grasp Part B, reorient Part B, and insert Part B onto Part C.

1.2 Difficulties

There are many inherent obstacles to a general and principled approach to manipulation problems. Articulated hands are redundant systems with many degrees of freedom. Exploiting this structure for choosing the best way to hold an object and then for altering its configuration within the grip is a complex process. Grasp planning is computationally expensive, the kinematics between the hand and the object are complicated, and the equations describing the manipulation within a

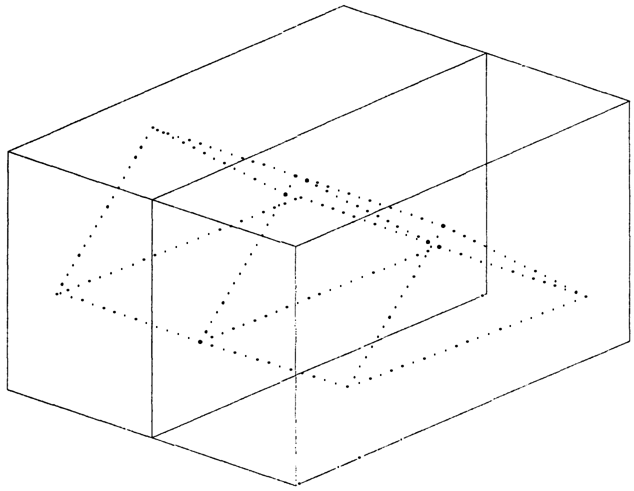

Figure 1.4: The complete assembly

grasp are differential, non-linear, time-varying, and constrained.

The *grasp finding* problem is a static problem: how to achieve a firm grip for arbitrarily shaped objects? This is an essential subproblem for manipulation because any activity between a robot hand and an object assumes the object to be held securely by the hand. The *manipulation problem* is a dynamic problem: how to affect changes to the configuration of an object held by a robot without losing the grip? This requires the continuous regrasping of the object. While there has been good progress for the static grasp planning problem, the dynamic regrasping planning problem has been left behind. The difficulty is in generating motion when there is an infinite number of admissible motions. There are many ways to go from one orientation of a grasped object to another. What is the best way and how should the robot find it? Finding the way to generate motion often involves the selection of intermediate goals, thus adding intricacy. The increased complexity in deciding the existence of motion plans for robot hands and in selecting them is largely due to the kinematics and dynamics of the interaction of objects in contact.

Yet another difficulty is that the inherent complexity of the problem must be tackled very fast. The hand, which is not allowed to lose its grasp of the object, achieves fine manipulation by fast, small, and precise finger motions. The robot has to sense the reaction of the grasped object to the finger forces, analyze it, issue the appropriate finger commands, and apply them in real-time.

A major trouble spot for the realistic analysis of the interaction between a robot and an object is the fundamental presence of *uncertainty* in the domain. Robot systems are subject to significant errors in control, sensing, and the geometric representation of the environment. The fingers of the robot hand can not be positioned on the surface of an object precisely. Exactness is also a problem for the control of the forces exerted by the robot and the sensed values. Furthermore, numerical inaccuracies are introduced by the finite-precision arithmetic that is used in the initial data, so the *a priori* knowledge of the geometry of the object is not exact. For coarse applications, such as spray painting, these are minor issues. But dexterous manipulation is fine manipulation through fine position and force control. Fine motion involves tactile sensing, force compliance, and detecting and compensating for errors, which are not negligible. Strategies that are guaranteed to succeed in spite of these limitations are required.

An algorithmic theory of manipulation based on high-level operations is needed for progress towards the autonomous behavior of robots. The fundamental property of dexterous robots is their capability to reorient objects in their grip. Manipulation is thus reorientation. A practical algorithm for reorientation must satisfy several properties. First, it must be general, in that it should be applicable to arbitrary objects for any reorientation. Second, since it is to be implemented on a real device it should involve simple finger motions that can be computed fast. Third, since the application domain is characterized by uncertainties which manifest themselves as imprecisions in calculations and inaccuracies in control, it should exhibit good stability properties.

1.3 Our Approach

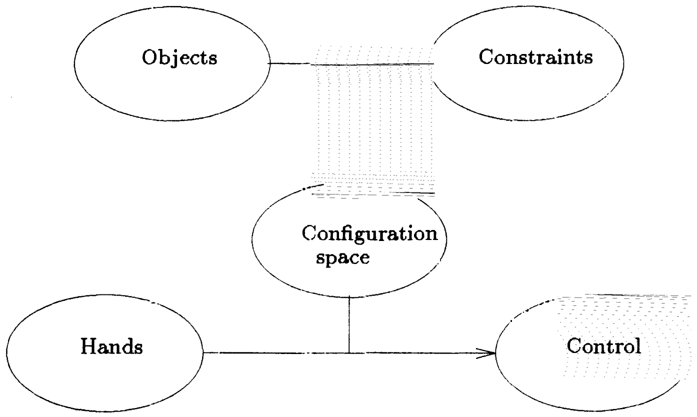

Figure 1.5: The manipulation framework

With the appropriate formal tools, the intricacies of the manipulation problem can be analyzed and solved. We propose a framework for manipulation that is theoretically well-founded and in which it is possible to use low-level tools systematically and effectively. This is accomplished using techniques from computational geometry, computational algebra, and computational mechanics. To bridge the gap from general motion planning to task-level planning, our approach is to treat the reorientation problem: for a given anthropomorphic robot hand, an arbitrary object, and a desired reorientation with respect to the hand, find an algorithm to synthesize a robust plan for the fingers that accomplishes the desired reorientation.

The framework is defined in terms of sets describing algebraically the manipulation quantities: the object, the robot hand, and the constraints on the motion. A manipulation algorithm is described as a function involving these sets (Figure 1.5). Since the interaction between the robot and the object is dependent on the geometric properties of the object and on the structure of the hand, we group the objects being manipulated in classes with common geometry, and the hands in classes with common structure. For instance, we have the class of polyhedra and the class of hands with n frictionless point fingers. A set of constraints on the contact between the hand and the object is included in the framework to serve as guides that enable compliance and facilitate the choice of manipulation strategies. A well-chosen set of constraints can improve greatly the performance of a manipulation strategy, as well as overcome some of the uncertainty.

The kinematics and dynamics of manipulation are characterized algebraically. This formulation allows the synthesis of ways to proceed from instantaneous analysis to large motions. Systems of equations are expressed and questions involving their solutions are answered by using the computational power of *Lie groups* and *Lie algebras*. Lie groups and Lie algebras form an area of mathematics in which modern algebra blends with classical analysis, and arise in the study of differential equations. They have played an important part in the development of certain parts of geometry. This has proven to be a computational tool of substantial power for manipulation systems, because it can characterize the geometry and physical properties of manipulation, and the continuity of motion. In addition to Lie algebras, we also use tools from computational geometry and automata theory to determine the properties that are useful for algorithmic simplifications.

The methodology employed to derive high-level manipulation strategies is based on functions relating the manipulation quantities and it is dependent on the role of the geometries of the object and the robot hand during reorientation. It is the interaction between the hand and the object that determines the type of primitive operations. Depending on how the grasping fingers constrain the motion of the object, we define two basic types of algorithms. The first type, called *finger tracking*, involves only a partial constraining on the motion of the object. Reorientations are generated by using the internal forces of the grip. The second type, called *finger walking*, requires total constraining of the motion of the object. The object is fixed with respect to the fingers of the grip and a reorientation step is accomplished by moving the fingers which carry the object. This type of algorithms is called finger walking. Both types of algorithms can be characterized in the framework we propose.

We have examined in detail how to proceed from instantaneous motion to large motions for finger tracking algorithms, for objects whose surfaces can be described algebraically and for robot hands with frictionless point fingers. An algorithm based on frictionless contacts is not general, but nonetheless, it has interesting features. For example, some application domains such as the manipulation under water or the cleaning of a spill, are characterized by such low coefficients of friction that the frictional forces can be ignored. We also show how to extend the analysis for robot hands with point contacts with frictions. The resulting algorithm is exact, and assumes perfect control. Uncertainty is fundamental for this application, but by developing an exact algorithm we gain insight into the problem, which eases the process of parameterizing it with the various factors of uncertainty. For the special class of planar rotations, the reorientation algorithm is robust in the presence of control and sensing uncertainties, and in the specifications of the initial data.

More specifically, we are given an object held by four frictionless fingers, whose locations are known. Three of the fingers stay fixed in space and the fourth is free to move. The object is constrained to be within the grasp of the four fingers at all time. When the fourth finger slides along its face applying normal forces, the object rotates. This process is called a *one-step* reorientation and the curve described by the sliding finger is called a *tracking curve*. The challenge is to determine the tracking curve that results in the desired rotational behavior for the object. Our approach is to describe the system of constraints and the laws of motion algebraically and to solve for the forces of the fingers and the tracking curve. This has proven to be a practical operation, since the resulting system is linear for any object whose surfaces can be described as time-varying functions. The implementation of the control for one-step tracking requires the ability of the robot to perform position control along the directions tangent to the surface of the object and force control along the normal direction. This is a feasible operation that can be accomplished by a hybrid controller [RC81,Mas81].

Several factors determine the duration of a one-step reorientation. The tracking motion terminates when a time interval elapses, when the force threshold exceeds a specified value, when one of the grasping fingers reaches an edge, or when the goal has been reached. The condition that determines the end of the one-step reorientation is called the *termination predicate*, an idea first introduced by [LPMT84]. To increase the range of orientations achievable by this technique, a strategy that alternates the tracking function between the four grasping fingers can be used. This results in multi-step algorithms.

Manipulation by finger tracking allows for the generation of arbitrary rotational motions of the grasped object. The key result supporting this is the description of the infinitesimal tracking motion as a linear system. The computations are simplified by suitably choosing a computational frame which is different than the task frame. When the motion of the object can be expressed as a sequence of simple time-varying functions, the algorithm can be used as a primitive. Of particular importance is the class of rotational motions whose axes remain constant over time. These are the planar rotational motions. This type of rotation is important in itself, since many fine assembly procedures require the planar reorientation of a part, for purposes of alignment. Moreover, any overall rotation can be obtained by composing such planar reorientations. Our algorithm provides a simple solution to the problem of controlling the planar rotational motions of a grasped object. We believe that this is an important primitive for high-level, task-directed, fine manipulation which can isolate the robot programmer from the complexity of low-level manipulation control.

The algorithm for maintaining planar motions can be applied to the three dimensional domain, as well as to the two dimensional domain. Of special interest is the case of reorientations in the plane. The functionality of this algorithm extends beyond manipulation with one anthropomorphic hand. An obvious extension is to use several cooperating robot hands. The algorithm can also be used for large scale manipulation with communicating mobile robots. The mobile robots emulate the roles of the fingers, by deciding among themselves which should have a constraining role and which should have the active role of the tracking finger. The significance of this application continues to increase due to the fast progress in designing mobile robots. Another domain of applicability is the area of manipulation with fixtures, which is receiving greater and greater attention for assembly line operations. For this direction, the fixtures have inherently the role of the fixed fingers.

For planar manipulation, we are given a simple polygon held by three frictionless fingers, whose locations are known imprecisely within some error. If two of the fingers stay fixed and the third one slides along an edge applying a normal force, then the polygon rotates. This is a *one-step* rotation. The special property of planar tracking is that there is a strategy to terminate each one-step rotation by using very simple sensing, achievable by an active slip sensor. This termination predicate is based on geometric properties and it involves getting the polygon *blocked* inside the grip. An active slip sensor reports termination when it detects that the object remains still in spite of continued displacement of the moving finger. This termination strategy is robust, in that it does not rely on accurate numerical calculations. Generating the finger motions only requires maintaining the current location in the case of the fixed fingers, or generating a compliant sliding motion for the moving finger.

We have developed a procedure to determine if polygons can be rotated arbitrarily, by associating a graph with the finger tracking algorithm and by searching it for cycles. For polygons that can be rotated arbitrarily, we determine *stability regions* for the robot fingers. These are intervals of the polygon with the property that when each finger originates anywhere within its interval arbitrary rotations for the polygon are guaranteed. Thus, the general procedure for the given polygon is: check if it can be rotated robustly and arbitrarily by building a small graph and searching for its cycles; if there is a cycle, determine the stability regions and proceed according to the motion plan.

It is important that a manipulation algorithm should be insensitive to small numerical perturbations and that it should not rely on accurate sensory feedback. Since such an algorithm is to be used in unstructured environments, it should operate on arbitrarily shaped objects. Since the manipulation is intended for a real robot, the finger motions that generate the desired manipulation should be computable in real time. The planar manipulation algorithm for rotations of polygons satisfies these properties.

1.4 Contributions

Dexterous manipulation is a special case of motion planning problems. Since the general motion planning with uncertainty problem was shown to be NEXP-hard [CR87], effort must be directed to defining classes of tasks that are tractable. We study the task of rotating polygons and polyhedra, and we show it is tractable and realistic, by providing a low complexity algorithm. For planar rotations, the algorithm is robust in the presence of uncertainties. This new strategy for reorientations is derived from geometric insights. A main feature is the ability to reason about the global behavior of the object being manipulated from the local data related to the direction of the finger forces. This results in surprisingly simple finger motions and a nearly sensorless strategy, which make possible our treatment of uncertainty. We also show that a finger motion plan based on blocking can accomplish arbitrary rotations for a class of polygons. An $O(n \log n)$ procedure to determine whether or not a polygon with n edges can be rotated in this way or not is provided, by building a graph with $O(n)$ edges associated with the geometry of the system and searching for a cycle. If there is no cycle, the maximal robust rotation for the given polygon can be determined. Furthermore, we calculate stability regions on the edges of such a polygon: intervals with the property that if each grasping finger originates anywhere within its interval, arbitrary rotations are guaranteed.

Our current work has taken steps towards building a framework for manipulation which is theoretically well-founded and simple to implement, and in which it is possible to use low-level tools systematically and effectively. This is a bridge from general motion planning towards achieving a task-directed approach to manipulation.

Current robot programming languages require the user to be aware of too much detail with respect to kinematics and control. Effort must be directed towards finding high level primitives. The introduction of finger tracking as a paradigm for dexterous manipulation is a contribution towards algorithmic, control, and language issues in robot programming.

1.5 Previous Work

Dexterous manipulation is a special but rich class of motion planning problems. Real achievements in this important area of robotics can be obtained only by interdisciplinary research. A number of advances in algorithmic and computational aspects of motion planning, grasping, manipulation, computational geometry, computational mechanics, and symbolic computation have contributed to the current state of the field.

1.5.1 Motion Planning

The general motion planning problem is the problem of finding a trajectory for a robot with any number of degrees of freedom, to move from a start configuration to a goal configuration, in an environment with obstacles. A thorough treatment can be found in [Lat91]. Path planning is intrinsically a computational geometric problem. Much work has addressed various restrictions for this problem. A collection of relevant papers is given in [HSS87,SY87]. An important advance for motion planning was the idea of representing the robot by its translational and rotational degrees of freedom in a configuration space [CS89].

An inherent characteristic of motion planning problems is the presence of uncertainties in the domain. The first attempt to directly incorporate uncertainty within planning was done in the seminal paper [LPMT84]. This work presents a theory for the automatic synthesis of motion plans in the presence of sensing and control uncertainties based on preimages. Preimages define a set of initial states. A motion from a preimage consisting of several atomic motions is guaranteed to reach the goal. This is recognized by the termination predicate. Although no effective procedure for computing preimages is proposed, a language for describing motion plans that our work also adopts is introduced. Preimages have been studies computationally in [Erd86]. In this paper, a simpler class of preimages called backprojections are proposed, as well as an algorithm to compute them. An efficient algorithm for computing backprojections was given in [Bri89]. Preimagens and backprojections were generalized to capture model error in [Don87a], and extensions to the backprojection algorithm were presented in [Lat89].

Several motion planning algorithms emerged in the framework of [LPMT84]. In [LP87] a motion plan for robots among polyhedra that is simple to implement and can be used for redundant manipulators was proposed, but the algorithm grows exponentially with the number of joints. The approach consists of using slice approximations in configuration space. This strategy speeds up the algorithm but it also causes it to miss paths for narrow environments. In [Don87b] a motion planning algorithm for a robot with six degrees of freedom that can rotate and translate in an environment with polyhedral obstacles is proposed. This solution is based on a search strategy that uses local information to reason about the global outcome. A different algorithm was proposed by [Buc89] for the synthesis of compliant motion strategies where the object slides along obstacles to reach the goal. Only translations are allowed, the task geometry is reduced to a finite state space, and planning is formulated as a best-first search. In [Can88] an exact algorithm for the general motion planning problem called the *roadmap* algorithm is proposed. This algorithm solves the problem in exponential time by using a coarse stratification on the configuration space and multivariate resultants.

The framework proposed by [LPMT84] is for building planners that are guaranteed to succeed. For some tasks, a guaranteed plan may not exist. Recognizing failures allows recovering from them and increases the scope of planning. This idea was recognized by [Don87a]. An algorithm for error detection and recovery strategies based on weak preimages was proposed. In [Erd89] the power of randomization for tasks with no guaranteed plans due to sensor uncertainties is addressed.

The complexity of the motion planning problem was first studied by Reif [Rei87]. This paper shows that the problem of planning a sequence of movements of linked polyhedra through three dimensional Euclidean space, avoiding contact with a fixed set of obstacles, is PSPACE-hard. The problem was studied further in [Nat88]. This work shows that motion planning for robots with uncertainty in sensing and control and a termination predicate with a sense of time is PSPACE-hard. For the particular class of cartesian robots, that is robots capable of realizing velocities only along the three axis of coordinates, the problem was shown to be PSPACE-complete. The general motion planning problem with uncertainty and the termination predicate proposed originally by [LPMT84] was shown to be NEXP-hard [CR87]. These complexity bounds are significant as they show that motion planning is a very hard problem. Effort must be directed towards finding suitable restrictions that can result in tractable algorithms for classes of objects. Our work follows this idea.

There are many other relevant contributions to motion planning, of which we note [BLP85,CD88,FHS88,FW88,HSS87,HW86a,HW86b,SY87,Xav92].

1.5.2 Grasp Planning

Of the specializations of the general motion planning problems, the two especially relevant to our work are grasp planning and manipulation planning. The question of how to achieve a firm grip for arbitrarily shaped objects is fundamental in the design and control of dexterous hands. Understanding grasping is an essential precondition for progress in dexterous manipulation, because any interaction between a robot hand and an object assumes that the object is held securely by the hand.

Previous work in grasping has been to analyze such aspects as grasp kinematics, the existence and stability of grasps, and synthesis methods for grasps with certain properties. Grasps have been analyzed with respect to properties such as equilibrium, stability, force closure, form closure, and positivity. Grasps have also been analyzed with respect to three types of contacts between the fingers and the object: frictionless point contact, point contact with friction (also known as hardfinger contact), and softfinger contact. Given some contact type, an important problem is to determine how many fingers are needed for a grasp with a specified property. For most grasp properties and contact modes, synthesis methods for the finger locations have been developed.

In [MP89] the problem of finding the number of fingers needed for a form closure grip of two dimensional and three dimensional objects is addressed. The main result is that form closure of a two dimensional object with piecewise-smooth boundary requires four fingers and that form closure of a three dimensional object without rotational symmetries can be achieved by seven fingers. This result also yields a notion of optimum stable grips and algorithms for efficient calculations of such grips. The optimality criterion is to minimize the grasping forces. In [BFG85] it was proven that evry polygonal object can be prehended stably with three finers so that its weight is balanced. In [JL87], a metric called the grasp goal function is introduced. This metric is used to define stable grasps and to derive an algorithm to locate such stable grasps. A different notion for grasp efficiency based on a quantitative form of Steinitz's theorem is given by [KMY92]. In [MSS87], the criteria for positive grips of polygonal objects is studied. Bounds for the number of fingers needed, and algorithms to synthesize such grips are presented. Grasp closure and stability were also analyzed in [Ngu88,Ngu89]. The main contribution of [Ngu88] is a simple algorithm for constructing force-closure grasps and for finding independent regions of contact for the fingers such that the motion of the grasped object is totally constrained. Analyzing whether a grasp with a total of n contacts, but requiring at least c contacts is force-closure can be done in $O(n)$. Synthesizing the n independent regions has complexity $O(cn^c 2^c)$ time. In [Ngu89] it is shown how all force-closure grasps can be made stable by constructing virtual springs at the contacts so that the grasped object has a desired stiffness matrix. We use this result to determine the initial grasps for our manipulation algorithm. An algorithm for planning robot grasping motions that are insensitive to bounded uncertainties in the object's location for parallel-jaw grippers was given by [Bro88]. This paper shows that by exploiting the task mechanics of squeezing, two degrees of uncertainty can be removed from the object's position. Squeeze-grasps, however, are insufficient for grasping all possible objects in the domain. In [Fea86] stable grasping when the object models are unavailable is analyzed.

A different method to account for uncertainty in grasping is given in [GM90, Gol90]. In this work, the probabilistic approach to grasp planning is analyzed.

There are manny other significant contributions to grasp planning, of which we note [PT89,HLMT90,Bas88,PS90,TLPM87,LS88,Fea86]. A good survey of grasping is given in [PT89].

1.5.3 Manipulation

Work has also been done towards understanding better the interactions between a robot hand and an object. Among the first contributions in the context of articulated manipulators was [MS85]. This work studies the mechanics for fingertip grasps for rigid body kinematics and Coulomb friction. A method is proposed to control the Salisbury hand to cause small arbitrary motions for the grasped object and to apply arbitrary forces. In [Cut85] the effects of the curvature of the finger tips and structural stiffness of the fingers are included.

Progress has also been made in the area of task-level mechanics. Here, the core is to study the interaction between the object and the robot. Pushing was one of the first strategies to be identified as useful for acquiring, reorienting, and repositioning objects. In [Mas86], the mechanics of planar pushing is analyzed. Theorems for predicting the motion of the object, based on a geometric analysis are developed. This effort results in strategies for manipulation with infinite planar manipulators.

The kinematics of contact that is, the motion of the point of contact over the surfaces of two contacting objects in response to some relative motion, has been examined in [Mon88]. The derived contact equations can be used to determine the curvature of an unknown object at a point of contact. The effects of friction on two objects in contact has been analyzed extensively in [Goy89].

In [EM88] the paradigm of sensorless manipulation is explored. The planner produces sensorless strategies which succeed in aligning or grasping an object whose initial configuration is not known. This strategy is applicable to a class of objects with no rotational symmetries about the center of mass.

The benefits of enveloping an object in the grasp of a robot hand and the way to achieve such grasps have been treated in [TAP88,TP90]. The manipulation of polygonal objects with a robot hand consisting of a planar palm with two hinged fingers to an enveloping grasp for which the object contacts the plane is addressed in [TAP88]. The final grasp is selected so as to minimize the contact forces. In [TP90] the problem of planning for manipulation with sliding contacts is explored. The approach is to manipulate an object to a secure and enveloping grasp by exploiting the structure of a manipulator with two sliding contacts. An initial grasp with which the object can be manipulated away from the support is first chosen. This grasp is then continuously altered until envelopment is achieved, under the quasi-static assumption. The role of finite slip between a grasped object and a robot hand has also been discussed in [Bro87].

In [Bro91] reliable manipulation tasks that involve two rigid polygonal objects interacting in the plane are investigated. The tasks considered include linear pushing, compliant motion, and placing-by-dropping. A collection of generic algorithms for the kinematic, static, and the dynamic analysis of these tasks is given.

The dynamics and kinematics for the infinitesimal motions resulting from the interaction between the robot fingers and the hand have been discussed in [LS88, LS89,Li89,LCS89]. In [LS89] an algorithm for coordinated finger motions based on the computed torque method is proposed. In [Li89] a framework for grasp planning, dexterous manipulation and coordinated manipulation based on infinitesimal motions is proposed. In this work, dexterous manipulation refers to the acquisition of a good grasp for a specific task, which is different than the meaning we associate, namely the ability of performing arbitrary reorientations. Motion planning for regrasping with rolling constraints is being examined.

There are other significant contributions to planning for manipulation, of which we note [HKK91,PS88,Nat89,AB89,Fea87].

1.6 Thesis Outline

This thesis is divided in 7 chapters and an appendix.

Chapter 2 presents the notational conventions adopted throughout this thesis. In this chapter we also introduce at a high level the concepts used for the formalization of this work. For the interested reader, the precise mathematical definition are included in Appendix A.

Chapter 3 presents a computational geometric view of planar reorientations by finger tracking. The multi-step reorientation algorithm is given and its robust termination condition is formulated. The algorithm is analyzed for generality and stability. A class of polygons that can be rotated arbitrarily and stably by the finger tracking algorithm with the robust termination condition is defined. The algorithm for planar manipulation has applicability to manipulation with robot hands, as well as large scale manipulation with mobile robots and manipulation with fixtures. This is a motivating example for the rest of the thesis.

Chapter 4 presents the general framework for dexterous manipulation. Dexterous manipulation is a geometric problem. We formulate it algebraically by defining systems of equations whose solutions are interpreted geometrically. The manipulation quantities and the relevant functions between them are introduced. The planar manipulation problem defined in Chapter 3 is reformulted in this framework, and new properties of the algorithm that result from the algebraic formulation are described.

Chapter 5 introduces finger tracking as a paradigm for dexterous manipulation. A detailed analysis of reorientations by finger tracking is given. Finger tracking can be used to brigde the gap between the low level analysis of the interaction between objects in contact and the high level actions for task-level manipulation.

Chapter 6 describes a simulator for the dexterous manipulation of polyhedra by finger tracking. The results of the simulation and their role in establishing the foundations for this problem are discussed.

Chapter 7 summarizes the conclusions of this work and indicates several direction for future investigations.

Chapter 2

Notation and Background

2.1 Notation

The following notation is adopted in this thesis:

1. Scalar quantities are denoted by Greek letters: α, β, *etc.*

2. Points in the plane are denoted by lower case letters: p, q, *etc.*

3. Vectors to points are denoted by boldface letters: $\mathbf{p}, \mathbf{q}$, *etc.*

4. Derivatives are denoted by $\dot{}$, such as $\dot{\mathbf{R}}, \ddot{\mathbf{R}}$, $\dot{\omega}$, *etc.*

5. Sets are denoted by upper case letters A, B *etc.* This includes line segments and polygons.

6. Families of sets are denoted by calligraphic letters $\mathcal{A}, \mathcal{B}$, *etc.*

7. Matrices are denoted by bold face upper case letters $\mathbf{R}, \mathbf{S}$, *etc.* A rotation matrix is denoted by $\mathbf{R} \in SO(n)$ and a translation vector is denoted by $\mathbf{u} \in R^n$. A Euclidean transformation is denoted by $\mathbf{T} \in SE(n)$, and its inverse is denoted by $\mathbf{S}$.

8. Coordinate frames are denoted by $\hat{x}, \hat{y}$ *etc.* The i, j, and k directions of the coordinate system $\hat{x}$ are denoted by $\hat{x}.x, \hat{x}.y$, and $\hat{x}.z$.

9. If $\mathbf{w} = (w_1, \ldots w_n)$ is a vector in some system of coordinates in R^n, its expression in homogeneous coordinates is denoted by $\overline{\mathbf{w}}$. To pass to homogeneous coordinates, let $\overline{\mathbf{w}} = (w_1, \ldots, w_n, 1)$.

10. The dot product of vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is denoted by $\langle \mathbf{v_1}, \mathbf{v_2} \rangle$.

11. The cross product of vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is denoted by $\mathbf{v_1} \times \mathbf{v_2}$.

12. The length of vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|$.

13. The determinant of the matrix determined by the vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ is denoted by $\det(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$.

14. The trace of matrix $\mathbf{A}$ is denoted by $\mathrm{tr}(\mathbf{A})$.

15. The robot hand is denoted by H.

16. The n fingers of the robot hand are denoted by $\mathbf{f_0}, \ldots, \mathbf{f_n}$.

17. The magnitude of the forces applied by the fingers of H is denoted by $f_0, \ldots, f_n$.

18. The object being manipulated is denoted by O.

19. The n vertices of the object are denoted by $\mathbf{p_0}, \ldots, \mathbf{p_n}$.

20. The normals of the faces of the object are denoted by $\boldsymbol{\nu}_0, \ldots, \boldsymbol{\nu}_n$.

21. The contact of the fingers on the faces of the object is denoted by $\mathbf{q_0}, \ldots, \mathbf{q_n}$.

22. The center of mass of O is denoted by $\mathbf{r_0}$.

23. The mass of the object is denoted by m.

24. The inertia tensor for O is denoted by $\mathbf{I}$.

25. The cartesian frame attached to O is denoted by $\hat{x}$ and the cartesian frame attached to H is denoted by $\hat{y}$.

26. $\mathbf{u}, \mathbf{R}, \dot{\mathbf{u}}, \boldsymbol{\Omega}, \ddot{\mathbf{u}}, \dot{\boldsymbol{\Omega}}$ denote the position, orientation, translational velocity, angular velocity, translational acceleration, and angular acceleration of the object.

27. $\mathbf{T}_t$ in $SE(3)$ is the matrix describing the position at time t of the object, relative to a frame of reference fixed on the robot hand. $\mathbf{S}_t = \mathbf{T}_t^{-1}$ is a matrix describing the position of the robot hand relative to a frame of reference fixed on the object. It follows that if $\mathbf{S}_t = \begin{pmatrix} \mathbf{R}_t & \mathbf{u}_t \\ 0 & 1 \end{pmatrix}$, then $\mathbf{T}_t = \begin{pmatrix} \mathbf{R}_t^{-1} & -\mathbf{R}_t^{-1}\mathbf{u}_t \\ 0 & 1 \end{pmatrix}$, where $\mathbf{R}_t \in SO(3)$ is a rotation and $\mathbf{u}_t \in R^3$ is a translation.

2.2 Background

Dexterous manipulation is the computational study of how robots affect the world around them by means of their end-effectors. This involves being able to guarantee a certain behavior as objects come in contact with robot hands. Understanding the interaction between the object and the robot is the quintessential component of dexterous manipulation. The study of motion without the consideration of the forces generating it is the subject of kinematics. The study of motion in connection with the forces and torques causing it is the subject of dynamics. In this thesis, we consider both the kinematic and dynamic aspects of manipulation. In this section we introduce the basic definitions related to the motion of a grasped object that will be assumed in the following chapters.

2.2.1 Rigid Motions

In general, R^n is the set of n-tuples of real numbers and it is called Euclidean n-space. A point in R^n is represented by its tuple of coordinates $x = (x_1, \ldots, x_n)$. An affine-linear transformation $f : R^n \rightarrow R^n$ is a continuous mapping defined by n-point-to-point linear transformations $f = (f_1, \ldots, f_n)$, where $f_i : R^n \rightarrow R, i =$

$1, \ldots, n$ such that for each $x \in R^n$, $f(x) = (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n))$, and

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= a_{1,1}x_1 \ldots a_{1,n}x_n + b_1 \\ f_2(x_1, \ldots, x_n) &= a_{2,1}x_1 \ldots a_{2,n}x_n + b_2 \\ \ldots &= \ldots \\ f_n(x_1, \ldots, x_n) &= a_{n,1}x_1 \ldots a_{n,n}x_n + b_n \end{aligned}$$

and $a_{ij}, b_j \in R$, $i, j = 1, \ldots, n$ are given. Let $\mathbf{A} = (a_{ij}), i, j = 1, 2, \ldots n$ be the matrix of the transformation f, let $\mathbf{b} = (b_1, \ldots, b_n)$ be the vector of free terms of f, and let $\mathbf{x} = (x_1, \ldots, x_n)$ be the vector defined by the coordinates of the point $\mathbf{x} \in R^n$. A linear transformation is thus represented by $f = \mathbf{Ax} + \mathbf{b}$.

The set of all non-singular linear transformations of R^n forms a group called *the general linear group*, and denoted by $GL(n)$. R^n may be given the n-fold product topology of R with itself. With respect to this topology, R^n is an affine space, preserving the notion of straight lines and parallelism between straight lines. The notion of distances between points is added by introducing metrics. R^n can be made into a metric space by defining the distance between two points as $\mathbf{x}, \mathbf{y} \in R^n$ by $\|\mathbf{x} - \mathbf{y}\| = \sqrt{(x_1 - y_1)^n + \ldots + (x_n - y_n)^n}$. This metric space is called the *Euclidean n-space.*

Definition 2.2.1 *A rigid motion is a homeomorphism of a Euclidean space onto itself, that preserves distances.*

In other words, a rigid motion is a map $f : R^n \to R^n$, such that $f(\mathbf{x}) - f(\mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$, for all $\mathbf{x}, \mathbf{y} \in R^n$.

Example 2.2.1 *Let* $\mathbf{u} \in R^n$. *The function* $f : R^n \to R^n$ *defined by* $f(\mathbf{x}) = \mathbf{x} + \mathbf{u}$ *is a rigid motion. This is a special motion called translation by* $\mathbf{u}$.

Example 2.2.2 *Let* $\mathbf{R} \in R^n \times R^n$, *where*$\mathbf{R}$ *is an orthogonal matrix. The function* $f : R^n \to R^n$ *defined by* $f(\mathbf{x}) = \mathbf{Rx}$ *is a rigid motion. This is a special motion called rotation by* $\mathbf{R}$.

It is well known from affine geometry that the collection of linear transformations is a group with respect to the usual operation of function composition. A very important subgroup of linear transformations is the group of rigid motions (also called Euclidean transformations). A different representation of rigid motions is provided by the following definition.

Definition 2.2.2 *An affine transformation $f : R^n \rightarrow R^n$ defined by*

$$f(\mathbf{x}) = \mathbf{R}\mathbf{x} + \mathbf{u}$$

is a Euclidean transformation if R is an orthogonal matrix.

Theorem 2.2.1 *(Euler): Every rigid motion is the composition of a rotation and a translation.*

Consider $SE(3)$, *the special Euclidean group*, defined by:

$$SE(3) = \left\{ \left(\begin{array}{cc} \mathbf{R} & \mathbf{u} \\ 0 & 1 \end{array} \right) \middle| \; \mathbf{R} \in SO(3), \mathbf{u} \in R^3 \right\},$$

where $SO(3)$ is called the special orthogonal group, and it is the group of 3×3 orthogonal matrices whose determinant is 1. If $\mathbf{x} \in R^3$ in some system of coordinates $\hat{x}$, and $\mathbf{S_1}, \mathbf{S_2} \in SE(3)$. Then $\mathbf{S_1}\overline{\mathbf{x}}$ is a particular rigid displacement from $\mathbf{x}$ to $\mathbf{R_1}\mathbf{x} + \mathbf{u_1}$. The effect of composing two such transformations $\mathbf{S_1}, \mathbf{S_2}$ is to displace $\mathbf{x}$ to $\mathbf{R_2}\mathbf{R_1}\mathbf{x} + \mathbf{R_2}\mathbf{u_1} + \mathbf{u_2}$, which is the same as first moving $\mathbf{x}$ to $R_1 x + u_1$ and then displacing this new location to $R_2(R_1 x + u_1) + u_2$. Thus, the group of rigid motions under composition is isomorphic to $SE(3)$ under matrix multiplication, and $SE(3)$ gives a matriceal representation of rigid motions.

2.2.2 Configuration Space

Given an object O, its location is described by an element of $SE(3)$. The translation component $\mathbf{u}$ defines its position. The rotation component $\mathbf{R}$ defines its orientation. Thus, the manipulation problem can be formulated as follows.

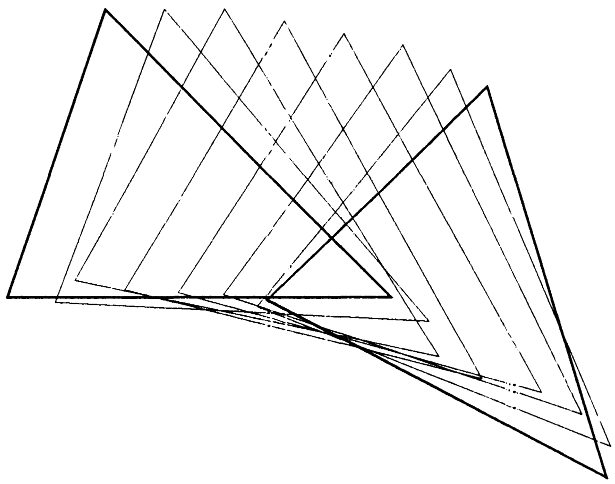

Figure 2.1: An example of manipulation

Definition 2.2.3 *Given an initial position and orientation of the object, and a final position and orientation of the object, the manipulation problem is to find a continuous sequence of positions and orientations that start at the initial location and end at the final location, and a set of finger forces that can generate the sequence.*

Figure 2.1 illustrates an instance of the problem for the manipulation of a triangle. The continuous sequence of positions and orientations is represented discretely.

Each instantaneous location of the object along the continuous path is specified in terms of a position and an orientation, relative to some reference frame. The position and orientation define the *configuration* of the object.

Definition 2.2.4 *The configuration space for an object is the set of all valid configurations.*

For a three dimensional object with 6 degrees of freedom whose motion is not restricted in any way, the configuration space is $R^3 \times SO(3)$, also represented by

$SE(3)$. In general, when the motion of a three dimensional object is constrained, the configuration space is a subset of $SE(3)$.

Example 2.2.3 *Let O be a planar object restricted to translate along the line given by $y = x$. The configuration space is $\{(x, y)|x, y \in R$ and $x = y\} \times SO(2)$.*

The path followed by an object being manipulated can be described by associating it with a curve in configuration space.

Definition 2.2.5 *A path of O from $\mathbf{q_{init}} \in SE(3)$ to $\mathbf{q_{final}} \in SE(3)$ is a continuous map $\tau : [0, 1] \to SE(3)$, such that $\tau(0) = \mathbf{q_{init}}$ and $\tau(1) = \mathbf{q_{final}}$.*

2.2.3 Trajectory Velocities and Accelerations

The understanding of the interaction between the robot and the object necessitates understanding the relation between paths in the configuration space and velocities and accelerations along the paths. The computational mechanism of Lie groups and Lie algebras provides clean descriptions of this relation.

$SE(3)$ and $SO(3)$ are classical examples of Lie groups. In this work, we will use their Lie group properties. A number of relevant definitions are in Appendix A. For a more detailed exposition, we recommend [AM77,Gil74,Ste65].

Lie groups and Lie algebras form an area of mathematics in which modern algebra blends with classical analysis, and arise in the study of differential equations. They have also played an important part in the development of certain parts of geometry, and vice versa, geometry has contributed to the development of Lie group theory. Since they are a natural tool for problems involving motion, they can be used as a computational tool for developing control algorithms for dexterous manipulation. We rely on the property that calculus on Lie groups is much simpler than calculus on manifolds.

$SE(3)$ and $SO(3)$ are inherent in the study of trajectories, and their Lie algebras are relevant in the study of velocities. The Lie algebra of $SO(3)$ is given by the group $Sk(3)$ of skew symmetric matrices, where $\mathbf{S} \in Sk(3)$ if $\mathbf{S} = -\mathbf{S}^T$. The Lie algebra of $SE(3)$ has the representation

$$\mathcal{L}_{SE(3)} = \left\{ \begin{pmatrix} \mathbf{S} & \mathrm{v} \\ 0 & 0 \end{pmatrix} \middle| \; \mathbf{S} \in Sk(3), \mathrm{v} \in R^3 \right\}.$$

The Lie algebra of $SO(3)$ has an important role in analyzing the velocity and acceleration constraints of rigid motion. In general, the velocity and acceleration of the motion of a body can be obtained either by direct differentiation, or by moving to the Lie algebra of $SE(3)$ and $SO(3)$. From a computational point of view, it is simpler to work with elements of $Sk(3)$ than to differentiate elements of $SO(3)$. For instance, if $\mathbf{Y}$ is the Lie algebra element corresponding to some rotation, the axis of rotation is defined by the three nonzero values defining the skew symmetric matrix $\mathbf{Y}$. The axis of rotation is less direct in $SO(3)$. If θ is a function of time and the rotation is:

$$\mathbf{R}_t = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

then the first and second derivatives are:

$$\dot{\mathbf{R}}_t = \dot{\theta} \begin{pmatrix} -\sin\theta & \cos\theta & 0 \\ -\cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{pmatrix}, \text{ and}$$

$$\ddot{\mathbf{R}}_t = \ddot{\theta} \begin{pmatrix} -\sin\theta & \cos\theta & 0 \\ -\cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{pmatrix} - \dot{\theta}^2 \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In terms of Lie algebras, with $\mathbf{Y}_t = \dot{\mathbf{R}}_t\mathbf{R}_t^{-1}$ and $\dot{\mathbf{Y}}_t + \mathbf{Y}_t^2 = \ddot{\mathbf{R}}_t\mathbf{R}_t^{-1}$, as shown by Proposition 2.2.2, we have:

$$\mathbf{Y}_t = \dot{\theta} \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \dot{\mathbf{Y}}_t = \ddot{\theta} \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The connection between trajectories and velocities at the Lie algebra level can be seen from the following well known result.

Proposition 2.2.2 *Let $\mathbf{Q}_t$ be a curve in $SO(3)$. $\mathbf{Y}_t = \dot{\mathbf{Q}}_t\mathbf{Q}_t^{-1}$ is a velocity curve in the Lie algebra of right invariant vector fields.*

Proof: The right translation by g is $R_g : G \to G$ with $R_g(h) = hg$. Since $R_{g_1}R_{g_2} = R_{g_1g_2}$ and $R_I = I$, we have that $R_g^{-1} = R_{g^{-1}}$. Now $R_g(I) = g$ so by the general principle that if $f(p) = q$ then $f_* : T_pM \to T_qN$, we have that $R_{g*} : T_IG \to T_gG$. The right translation maps the tangent space at the identity to the tangent space at g. Then $R_{g^{-1}*} : T_gG \to T_IG$. Now if $\mathbf{Q}_t \in SO(3)$, then $\dot{\mathbf{Q}}_t \in T_{\mathbf{Q}_t}SO(3)$, which implies that $R_{\mathbf{Q}_t{}^{-1}*}\dot{\mathbf{Q}}_t \in T_ISO(3)$, and so $\dot{\mathbf{Q}}_t\mathbf{Q}_t^{-1} \in T_ISO(3)$. It is easy to see that $\dot{\mathbf{Q}}_t\mathbf{Q}_t^{-1}$ is skew symmetric since $\mathbf{Q}_t \in SO(3)$ implies that $\mathbf{Q}_t^T\mathbf{Q}_t = \mathbf{I}$. By derivation, it follows that $\dot{\mathbf{Q}}_t^T\mathbf{Q}_t + \mathbf{Q}_t^T\dot{\mathbf{Q}}_t = \mathbf{0}$, and so $\dot{\mathbf{Q}}_t^T\mathbf{Q}_t + \mathbf{Q}_t^{-1}\dot{\mathbf{Q}}_t = \mathbf{0}$, or $\mathbf{Q}_t^{-1}\dot{\mathbf{Q}}_t = -\dot{\mathbf{Q}}_t^T\mathbf{Q}_t$. This shows that $(\mathbf{Q}_t^{-1}\dot{\mathbf{Q}}_t)^T = -\mathbf{Q}_t^{-1}\dot{\mathbf{Q}}_t$. □

2.3 Summary

In this section we have defined the notation used in this thesis. We have also introduced some basic motion planning concepts that will be referred to in the next chapters.

Chapter 3

Planar Manipulation

In this chapter we introduce the paradigm of constraint-based manipulation by finger tracking, we describe an efficient algorithm for the reorientation of polygons with anthropomorphic robot hands, and an efficient procedure for whether a polygon can be reoriented arbitrarily. This algorithm will be shown to have good stability properties, in that it is not sensitive to small errors in the initial data and it relies on a simple control scheme. This algorithm results from an *ad-hoc* geometric analysis, and it serves as a motivating example for the methodology for manipulation proposed in this thesis.

We assume robot hands with four fingers whose finger tips are frictionless points. For simplicity of exposition, we first show and analyze the algorithm for the class of triangles, in Section 3.1. Although the this is a simplified case, it captures the main ideas for the general case. In Section 3.2 we show and analyze the extension to polygons.

3.1 Rotating triangles

Let T be a triangle with vertices A, B, and C. Let H be a robot hand with three frictionless point fingers. Each finger exerts a normal force on a different edge of the triangle and it is represented by $\mathbf{f_i}$, $i = 0, 1, 2$. The contact points of the fingers are denoted by P, Q, and M. Let $\hat{x}$ be a global system of coordinates. In this coordinate system, the points C, B, A, P, Q, M correspond respectively to the vectors $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$. Without loss of generality, we assume only clockwise rotations.

3.1.1 One-step Rotations

Each finger exerts a normal force on a different edge of the polygon. The fingers have a good grip on the object if they satisfy the force closure property [Ngu88].

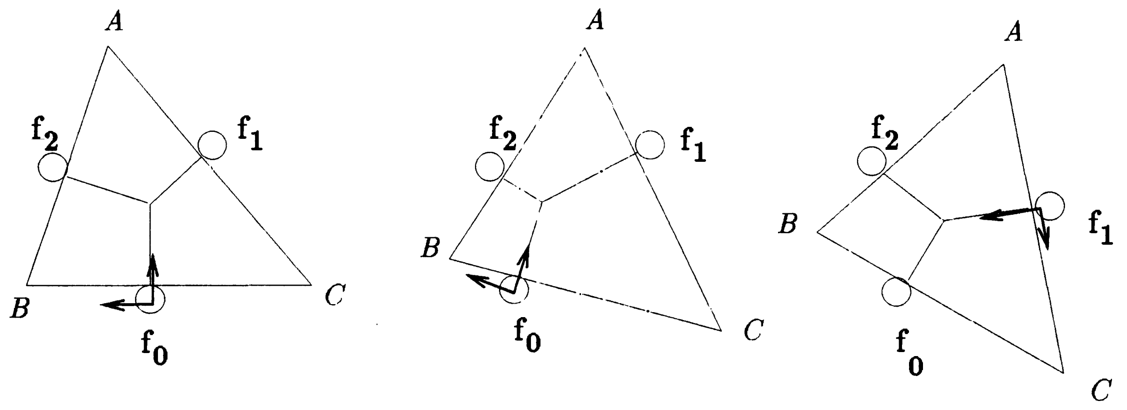

Figure 3.1: The reorientation of a triangle

Definition 3.1.1 *A grasp is called force closure if the fingers can resist any force applied on the object. The geometric condition for a force closure grasp is that the force directions meet at a point. In addition, none of the three angles detemined by the three pairs of force directions should exceed π.*

The point of concurrency of the forces is called the *center of grip*. All three configurations shown in Figure 3.1 are force closure.

The following observation leads to a simple reorientation algorithm. Consider an object grasped by three forces in a force-closure grasp. Keep two of the fingers

Notation: T is the triangle and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$ the fingers of the hand.

θ is the desired reorientation.

Input: θ.

Initialization: Grasp T using $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$, satisfying force-closure property.

Repeat: push with sliding contact with $\mathbf{f_2}$ keeping $\mathbf{f_0}$ and $\mathbf{f_1}$ fixed

until the termination condition occurs

Figure 3.2: Algorithm 3.1 for One-step Reorientation

fixed in the plane and constrain the object to be in contact with them. If the third finger is displaced so that its applied force does not pass through the intersection point of the other two fingers, the object rotates until the forces become concurrent. This result is quantified in [Mas86,Rus90].

Theorem 3.1.1 *Assume an object is being pushed with sliding contact. Then if the instantaneous rotation center lies to the right of the line of force, clockwise rotations occur. If the instantaneous rotation center lies to the left of the line of force, counter-clockwise rotations occur. If the rotation center lies on the line of force, translations occur.*

Suppose now that instead of applying a discrete force displaced from the center of grip, we start with a force-closure grasp on the triangle and one of the fingers moves continuously on its edge, applying normal forces. The object undergoes a continuous rotation according to Mason's theorem, until the three force directions become concurrent. This suggests a method for generating rotations, which is described by Algorithm 3.1 and depicted in Figure 3.2.

In Algorithm 3.1 fingers $\mathbf{f_0}$ and $\mathbf{f_1}$ stay fixed in the plane. They have the role of constraining the motion of the triangle, which is a passive role in the rotation phase. We call them *fixed fingers*. Finger $\mathbf{f_2}$ pushes with sliding contact to generate the reorientation relative to the other fingers, and thus its role is active. We call this type of finger a *tracking finger*. The three edges on which the fixed and tracking fingers are acting form the *grasping edge set*.

Definition 3.1.2 *The process by which a finger applies a force while sliding on an edge, thus causing the grasped object to rotate, is called one-step finger tracking.*

Figure 3.1 shows two one-step reorientations of a triangle. The second configuration is obtained by keeping fingers $\mathbf{f_1}$ and $\mathbf{f_2}$ fixed and by tracking with $\mathbf{f_0}$. The third configuration is obtained from the second, by keeping fingers $\mathbf{f_0}$ and $\mathbf{f_2}$ fixed and by tracking with finger $\mathbf{f_1}$.

Algorithm 3.1 describes loosely a scheme for object reorientation. Its further refinment leads to a precise, implementable, and provably correct strategy. Its structure consists of the initialization step, the repeated finger tracking action, and the termination condition. Of these steps, the initialization can be defined by using efficient existing algorithms for force-closure grasp synthesis [Ngu88]. The exact meaning of the repeated action and of the termination condition however, should be specified. In addition, a number of issues should be resolved for a complete understanding of reorientation by finger tracking. Is this operation feasible? Is there a precise geometric characterization for the motion of the object and the tracking finger? What is the relationship between the path of the tracking finger and the path of the object? What are the limitations of the one-step rotation? Can these limitations be overcome to yield an algorithm that guarantees arbitrary rotation? How stable is this algorithm? Is there a termination predicate that is insensitive to small errors in the initial data?

Before analyzing the questions that address the global problem – that is, why Algorithm 3.1 works – the physical realizability of finger tracking should be resolved. A tracking finger necessitates the ability to control applying a force and sliding compliantly. Such a composed operation can be achieved by a hybrid controller [Mas81,RC81] operating in local coordinates. The tracking operation can be generated by position-control in the directions tangential to the surface of the object, and force-control in the normal direction. This local control can be transformed into global control by using the matrix relating the local and global frames. The following sections discuss the remaining issues.

3.1.2 Analysis of One-step Rotations

The basis for why Algorithm 3.1 generates rotations is Mason's theorem that relates a pushing force and the instantaneous center of rotation to the resulting motion. A characterization for the path of the instantaneous center of rotation for the object-hand system is needed. In this section we describe the geometric properties of finger tracking that concern the path of the instantaneous center of rotation, the path of the triangle, the path of the tracking finger, and the relationship between them and the overall rotation.

Proposition 3.1.2 *The locus of the vertex determined by the edges in contact with fixed fingers is a unique circle. The locus of the center of grip is the same circle.*

Proof: We show that the vertex A and the center of grip G rotate on the unique circle defined by A, P, G, and Q. The quadrilateral $APGQ$ can be inscribed in a circle, because $\angle(AP, PG) = \angle(AQ, QG) = \pi/2$. Now consider displacing T to the arbitrary position T'. The new vertices are A', B', and C' and the displacement is done subject to the constraint $P \in A'B'$ and $Q \in A'C'$. The new grip center is G'. Observe that the circle defined by A, P, G, Q is the same as the circle defined by A', P, G', Q, because $\angle(BA, AC) = \angle(B'A', A'C')$ since they both measure half of the length of the circular arc PQ. In other words, if we look at an arbitrary position $A^iB^iC^i$ with $P \in A^iB^i$ and $Q \in A^iC^i$, the A^i vertex is on a fixed circle and its position on the circle determines uniquely the position of the triangle. By the same argument, the center of grip G^i lives on the same circle, and its position is diametrically opposed from A^i. Moreover, when vertex A moves, it sweeps continuously an arc of this fixed circle and so does G. □

Proposition 3.1.2 gives a simple geometric characterization for the motion of a selected feature of the triangle. Although this is a local property, it has global consequences. In particular, it shows that a triangle whose motion is constrained at two points on separate edges is a one-degree of freedom system. In addition, the evidence that a specific vertex moves according to a circular path facilitates measuring the global reorientation of the object. Each position of the designated vertex corresponds to a different orientation of the triangle. There is a connection between the change in the orientation of the triangle and the change in the position of the two selected points on the circle.

Corollary 3.1.3 *When the vertex determined by the edges in contact with fixed fingers (or, equivalently, the center of grip) sweeps an arc of length α, the change in the orientation of the triangle is $\frac{\alpha}{2}$.*

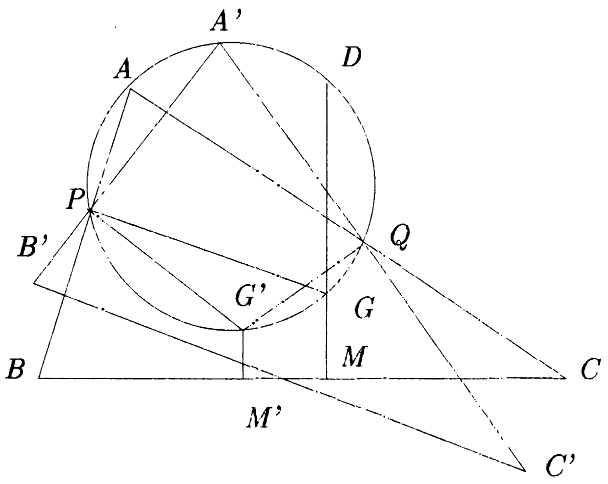

Figure 3.3: Geometric properties of finger tracking

Proof: Let O be the center of the circle in Proposition 3.1.2. If $\angle(AO, OA') = \alpha$, then $\angle(A'P, PA) = \alpha/2$. Since P is a fixed point in the plane, it follows that the motion of the triangle obtained by rotating A on its circle by α is equivalent to a rotation of the triangle about O by $\alpha/2$, composed with some fixed translation (which depends on the position on A and on α). But the composition of a rotation and a translation is a rotation through the same angle as the first rotation, with a different and uniquely determined center. Thus it follows that the motion of T, from a start point to a final point, is equivalent to a rotation by $\alpha/2$ with respect to some uniquely determined point. Here, equivalence means that there is such a rotation that takes the triangle from the same start position to the same final position, but following perhaps a different path. □

The shape of the path of the vertex determined by the edge of the two fixed fingers is only a local property. Not all the points of the triangle move along circles. An insight into the global motion is given next.

Proposition 3.1.4 *The instantaneous center of rotation is the center of the grip.*

Proof: The motion of the triangle from A to A' is equivalent to a rotation about some point R_c, as observed above. Then the following segments have equal lengths: $R_cB = R_cB', R_cA = R_cA'$ and $R_cC = R_cC'$. Since the triangle is rigid and the amount of rotation is $\alpha/2$, it follows that $\angle(AR_c, R_cA') = \angle(BR_c, R_cB' = \angle(CR_c, R_cC') = \alpha/2$. So R_c is the intersection of the perpendicular bisectors of AA', BB' and CC'. Then R_c is defined by the intersection of the perpendicular bisector of the chord AA' and the circle defined by the points A, P, G, and Q, because the circular arc AA' has length α and the amount of rotation is $\alpha/2$. When $A' \to A$ the perpendicular bisector becomes a diameter. In other words, R_c diametrically opposes A. But the center of the grip diametrically opposes A according to Proposition 3.1.2. Since G varies on the fixed circle, it follows that the triangle moves such that the instantaneous center of rotation sweeps an arc of the same circle. □

Since the center of grip is not fixed, but moves along a circle according to Proposition 3.1.2, the instantaneous center of rotation changes continuously. Although this shows that the curve for the global motion of the object is complicated, it also enables a precise specialization of Mason's pushing theorem to this system.

Theorem 3.1.5 *Given a triangle gripped by three point fingers, two of which providing fixed point constraints, the motion of the object is unique. Let G be the point determined by the force directions of the fixed fingers. Then the object does not move if the force direction of the tracking finger contains G, it rotates clockwise if the force direction of the tracking finger lies to the left of G, and it rotates counterclockwise otherwise.*

Proof: It is easy to see that the triangle stays fixed in the first case, because the given condition is precisely the condition of *force closure.* For the second case, we use the result from [Mas86], which states that if the force is to the right of the instantaneous center of rotation, *i.e.* G, then the rotations are counterclockwise, and if the force is to the left of the instantaneous center of rotation, then the rotations are clockwise.

Since the vertex of the triangle and the intersection point of the force directions sweep circular arcs continuously, the motion resulting from pushing with $\mathbf{f_2}$ at a fixed point X on the edge is equivalent to the motion resulting from slowly sliding $\mathbf{f_2}$ towards X, until $\mathbf{f_2}$ reaches X. □

The properties described thus far are consequences of the kinematic constraints. Another essential ingredient of the problem is the presence of the tracking finger, which can slide anywhere on its edge. It is the behavior of this finger that affects the orientation of the object. For motion planning applications, the path of this finger and the dependency of the object orientation on points along this path are needed.

Proposition 3.1.6 *The locus of the tracking finger is a fixed circle.*

Proof: Let h_A be the altitude of the triangle and let D be the intersection of the force direction $\mathbf{f_0}$ with the circle through A, P, G, Q. We will show that the locus of the moving finger M is the circle of center D and radius equal to the length of h_A. Notice that $\angle(GD, DA) = \angle(AQ, QG)$ and since GQ is perpendicular on AC it follows that $\angle(GD, DA) = \pi/2$. Let AA^h be the altitude of the triangle at A. It now follows that $ADMA^h$ is a rectangle, which implies $AA^h = DM = h_A$. In other words the length of the perpendicular at M between M and the circle through A, P, G, Q is constant. So in order to complete the proof we have to show that for any position of the triangle, say $A'B'C'$, subject to the given constraint, the intersection between $M'G'$ and the circle through A, P, G, Q is precisely D. To see that M', G' and D are collinear, let us evaluate their angle. We have that $\angle(M'G', G'D) = \angle(M'G', G'P) + \angle(PG', G'D)$. But $\angle(PG', G'D) = \angle(PQ, QD) = \angle(DG, GP)$. Since DG is perpendicular on BC and GP is perpendicular on AB we get that $\angle(PQ, QD) = \angle(AB, BC)$. We also have that $\angle(M'G', G'P) = \pi - \angle(PB', B'M') = \pi - \angle(AB, BC)$, because the quadrilateral $PB'M'G'$ can be inscribed in a circle. By putting all the pieces together, we get that $\angle(M'G', G'D) = \pi - \angle(AB, BC) + \angle(AB, BC)$, hence M', G', and D are collinear. □

Corollary 3.1.7 *The motion of the tracking finger contact measures the rotation of the triangle with respect to the fixed finger contacts.*

Proof: If the length of arc MM' is α, then $\angle(BC, B'C')$ is also α because DM' is perpendicular on $B'C'$ and DM is perpendicular on BC. □

The main limitation of a one-step rotation is that there is a bound on the maximum reorientation angle. Since the fingers must keep a continuous grasp on the triangle, the tracking process can proceed only until one of the fingers reaches a vertex.

Proposition 3.1.8 *The maximum reorientation of a one-step rotation is $\frac{\pi-\alpha}{2}$, where α is the measure of the vertex determined by the edges of the fixed fingers.*

Proof: The angle $\frac{\pi-\alpha}{2}$ is the measure between the two extreme positions of the triangle, namely the position where vertex A coincides with P and the position where vertex A coincides with Q. This maximum is not attainable when the distance between the fixed fingers is larger than one of the triangular edges. □

3.1.3 Multi-step Rotations

Notation: T is the triangle and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$ the fingers of the hand.

θ is the desired reorientation and r the accomplished rotation.

t denotes the index of the tracking finger.

Input: θ.

Initialization: Grasp T using $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$, satisfying closure property.

$r = 0; t = 0; n = 0$

While $r \neq \theta$ **do**

begin

one-step tracking: finger $\mathbf{f_t}$ tracks

fingers $\mathbf{f_i}, i \neq t$ are fixed

until $\lambda(\mathbf{p_0}, \ldots, \mathbf{p_2}, \mathbf{f_0} \ldots \mathbf{f_2})$

update r; $n = n + 1$; $t = (n + 1) \bmod 3$

end

Figure 3.4: Algorithm 3.2 for Multi-step Finger Tracking

The factors that limit the total rotation attainable in a one-step rotation are the geometry of the object and the initial grasp. It is possible to increase the maximum rotation by composing one-step rotations in a multi-step algorithm. The transitions between two one-steps is done by reassigning the constraining and the tracking functions to the three grasping fingers. Algorithm 3.2 for multi-step rotations is described in Figure 3.3.

The termination predicate for each one-step tracking is given by a fuction $\lambda(\mathbf{p_0}, \ldots, \mathbf{p_2}, \mathbf{f_0}, \ldots, \mathbf{f_2})$ which depends on finger positions and forces. This condition is defined by some chosen property, such as requesting that no finger should fall off its edge, no finger should be any closer than ϵ from the vertices of the object, no finger should move closer than half of its current distance to a vertex, *etc.* Each termination condition yields a different *protocol* for the finger tracking algorithm.

Proposition 3.1.9 *The reorientation accomplished by the multi-step tracking Algorithm 3.2 is equivalent to the reorientation accomplished by a revolving wrist.*

Proof: Since each finger is fixed on an edge, when an edge has completed a 2π rotation, so has its corresponding finger. Thus, the motion given by Algorithm 3.2 is equivalent to the motion performed by a revolving wrist. □

The fingers of a mechanical hand have limited reachability. A simple change in Algorithm 3.2 models this property. The new algorithm involves four fingers. At each point in time, three of the four fingers are holding the triangle and of the three fingers, two are fixed and one is tracking. The fourth finger is *free*. The rotation step proceeds as in Algorithm 3.2. When the termination condition of the tracking step occurs, the free finger becomes a tracking finger and one of the previously fixed fingers becomes free. This scheme limits the overall motion of the fingers. The algorithm is given in Figure 3.4.

The selection of the new free finger in Algorithm 3.3 is done so as to preserve the grasp and to avoid collisions between the fingers. A simple strategy is to always preserve the relative order of the fingers.

Notation: T is the triangle and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$ the fingers of the hand.

θ is the desired reorientation and r the accomplished rotation.

t denotes the index of the tracking finger.

Input: θ.

Initialization: Grasp T using $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$, satisfying the force closure property.

$r = 0; t = 0; n = 0; l = 3$

While $r \neq \theta$ **do**

begin

track using finger $\mathbf{f_t}$, keeping $\mathbf{f_i}, i \neq t, l$ fixed, and $\mathbf{f_l}$ free

until $\lambda(\mathbf{p_0}, \ldots, \mathbf{p_3}, \mathbf{f_0}, \ldots, \mathbf{f_3})$

update r; $n = n + 1$; $t = l$; $l = (l + 1) \bmod 3$

end

Figure 3.5: Algorithm 3.3 for Multi-step Finger Tracking

3.1.4 Achieving Arbitrary Rotations

A general algorithm for dexterous reorientations allows for arbitrarily large rotations. The ability of doing rotations larger than 2π is useful for assembly tasks such as joining two parts together with a wing-nut or a hex-nut. In this section we show that there are finger tracking protocols that can perform infinitely large rotations. We note that talking about rotation relative to fixed fingers is equivalent to talking about how the finger contacts move on their edges.

Theorem 3.1.10 *Every triangle can be rotated arbitrarily by a finger tracking protocol.*

We prove this theorem by showing the existence of a cycle of finger tracking steps.

Definition 3.1.3 *We say that we have a cycle of n finger tracking steps if we start from some initial grasp and after n steps, the final grasp coincides with the initial grasp. Then the sequence of rotations forms an n-cycle.*

Lemma 3.1.11 *For every triangle, there is a starting configuration that gives a 3-cycle.*

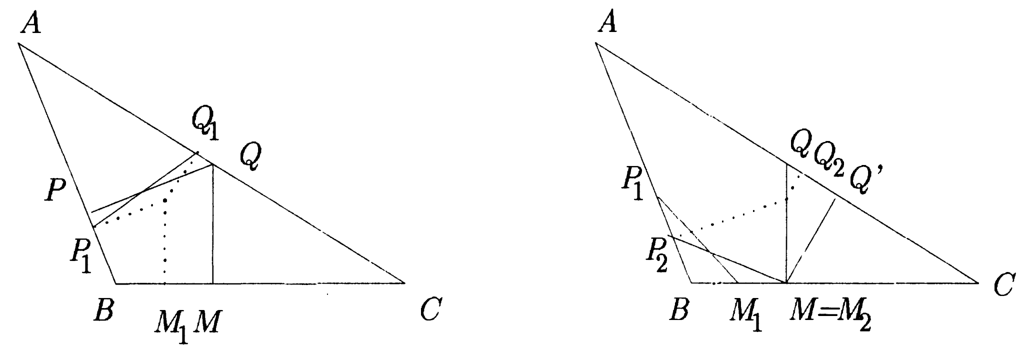

Figure 3.6: The construction of a cycle

Proof: We prove the lemma for the harder case of obtuse triangles. For acute triangles we give a special 3-cycle in the next section. Let M, P, and Q be the initial finger contacts, such that APQ is a right angle triangle with the center of grip at Q. Note that $PQ \perp AB$ and $QM \perp BC$. We show that there is a sequence of rotation angles α, β, and γ that brings the finger contacts to the initial positions.

In the first step, track with the finger at M to generate a rotation $\alpha > 0$. Let M_1, P_1, and Q_1 be the new finger positions. Since $CQ < CQ_1, CM < CM_1$, and $QM \perp BC$, it follows that $QM < Q_1M_1$. Choose α so that $P_1M_1 > BM$. Note that this is always possible because for $\alpha = 0$, $P_1M_1 = PM > BM$ and as α increases, the length of P_1M_1 decreases in a continuous fashion.

In the second step track with the finger at Q_1 and let P_2, M_2, and Q_2 be the new finger positions. Consider the case when M_2 coincides with M (Figure 3). Since $P_1M_1 > BM$ and the distance between the fixed fingers remains constant, it is always possible to track until M and M_2 coincide. For this configuration, P_2 is between P and B, and thus Q_2 is between Q and Q', where $MQ' \perp AC$. This implies that $Q_2M_2 \leq QM$, so $Q_1M_1 > QM > Q_2M_2$. Since the length Q_2M_2 changes in a continuous manner, there is a rotation β such that $QM = Q_2M_2$.

In the third step, track with the finger at P_2 to bring the fingers to their initial locations. The distance between the fixed fingers is $Q_2M_2 = QM$ and thus there is a rotation angle γ that completes the 3-cycle. Observe that for each triangle there is an infinity of starting configurations that give 3-cycles. □

3.1.5 Achieving Arbitrary Robust Rotations

The termination predicate $\lambda(\mathbf{p_0}, \ldots \mathbf{p_3}, \mathbf{f_0}, \ldots, \mathbf{f_3})$ is the critical part of Algorithm 3.2. The weakest termination condition for each one-step tracking is that the grasp be preserved – that is, no finger should move past a vertex. By using the geometric properties described in Section 3.1.2, it is possible to calculate the amount of tracking that takes a finger contact to a vertex. This approach is prone to failure because of numerical inaccuracies, introduced either in the finite-precision arithmetic that is used, or in the initial data. Even if the termination condition is strengthened by requesting that no finger should ever be closer than ϵ from a vertex, determining the smallest ϵ involves fragile numerical calculations.

A different strategy is to look for termination predicates that do not depend on exact *a priori* knowledge of the geometry of the object and location of the fingers, but rely instead on properties detectable by simple sensing. A termination predicate of this nature is given next.

Proposition 3.1.12 *Suppose that during a step the tracking finger moves along the edge $E = (v_1, v_2)$ towards vertex v_1. If the center of grip becomes identical* *to any fixed finger contact, then the motion is blocked, i.e. no furter rotation can result if the tracking finger proceeds towards* v_1.

Proof:

Let ABC be a triangle as before, with $\angle A$ acute and let $\mathbf{f_3}$ be the finger pushing on BC. We show that if P and Q are such that $| PQ |< h_B$ and $\mathbf{f_3}$ causes clockwise rotations, or if P and Q are such that $| PQ |< h_C$ and $\mathbf{f_3}$ causes counterclockwise rotations, then the triangle reaches a configuration in which it is blocked, *i.e.* $\mathbf{f_3}$ can no longer cause motion by tracking.

Without loss of generality, assume that $\mathbf{f_3}$ tracks towards B, that is, it causes a clockwise motion on the triangle. The fact that the triangle reaches a blocking configuration means that the triangle reaches a state in which $\mathbf{f_3}$ can no longer cause any clockwise motion by pushing. In other words, in order to continue the constraint motion, a force component in the direction opposite to $\mathbf{f_3}$ is needed.

Since the contacts at P and Q are frictionless, $\mathbf{f_2}$ and $\mathbf{f_3}$ can cause no virtual work. Now based on the given constraints, we can see that the motion of every point on the segment BM is a portion of a limaçon. Furthermore, when PQ becomes perpendicular on AC these points attain a local extreme in the direction along $\mathbf{f_1}$. If the motion were to continue past this stage, it would mean that $\mathbf{f_1}$ has a negative virtual work (since the motion is in the opposite direction of $\mathbf{f_1}$) which is impossible. Therefore, the blocking happens when PQ is perpendicular on AC. □

For a geometric interpretation, consider a clockwise labeling of the edges of the polygon involved in the grasp. The polygon is in a blocked grasp for tracking on edge i if the line between the fixed fingers is perpendicular on the predecessor of i. This can only happen when the angle determined by the edges assigned to fixed fingers is acute.

Figure 3.7 shows a blocked triangle. The center of the grip is on edge AB and the triangle is blocked for clockwise, but not for counterclockwise rotations. We

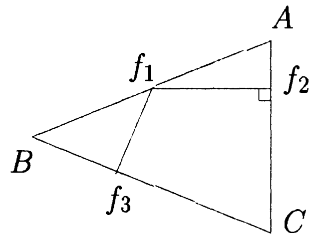

Figure 3.7: A blocked triangle

say that the condition of Proposition 3.1.12 is the *blocking condition* and the finger locations for a blocked object form a *blocked configuration.*

In situations where blocking occurs, there is no need to precalculate the finger motions to end a one-step rotation. The tracking finger could do its pushing/sliding motion controlled by a hybrid controller, until an active slip sensor detects if the object is blocked inside the grip.

Definition 3.1.4 *A termination predicate* $\lambda(\mathbf{p_0}, \ldots \mathbf{p_3}, \mathbf{f_0}, \ldots, \mathbf{f_3})$ *that is constant in an entire neighborhood of* $\mathbf{p_0}, \ldots \mathbf{p_3}, \mathbf{f_0}, \ldots, \mathbf{f_3}$ *is called robust.*

A robust termination predicate does not rely on accurate calculations and initial data. The edge intervals for the initial finger locations that yield the same robust behavior are called *regions of stability.*

It is intuitively clear that for a one-step tracking of an acute triangle Proposition 3.1.12 provides a robust termination predicate. As long as the distance between the fixed fingers is smaller than the smallest altitude, the hypothesis of Proposition 3.1.12 is achievable. For this to happen, the only requirement is that the distance between the fixed fingers be smaller than some value.

We now use Proposition 3.1.12 to first exhibit a very special 3-cycle for acute-angled triangles. This is a cycle of three blocked configurations. We then show that this cycle is stable, that is, small imprecision in the positioning of the fingers still results in cyclic behavior when following the finger-tracking algorithm. We also compute the degree of imprecision tolerable by the cycle.

Proposition 3.1.13 *Let PQM be a blocked grasp obtained by tracking on AB with $PQ < h_B$, the altitude at B. Then tracking on BC results in a blocked grasp $Q'P'M'$ where $AP' = AQ$.*

Proof: Since $PQ = P'Q'$ and $\angle(AP, PQ) = \angle(AQ', Q'P') = \frac{\pi}{2}$ triangles APQ and $AQ'P'$ are congruent. Hence $AP' = AQ$. $\square$

We use this observation to first construct the 3-cycle of blocked rotations for acute-angled triangles.

Theorem 3.1.14 *Every acute triangle has a 3-cycle of blocked rotations.*

Proof: Consider Figure 4. We observe that if we find three points $Q \in AC$, $P' \in AB$ and $M'' \in BC$ such that $AP' = AQ$, $BP' = BM''$ and $CQ = CM''$, then starting from the configuration PQM we have a 3-cycle of blocked rotations, where $QP \perp AB$ and $QM \perp BC$.

To see this, observe that finger-tracking on BC results in blocked configuration $M'P'Q'$ where $P'Q' \perp AC$, $P'M' \perp BC$ and $PQ = P'Q'$. From the observation above, this implies that $AP' = AQ$. Similarly, finger tracking on AC will result in the blocked configuration $Q''M''P''$ where $P'M' = P''M''$. This means that $BP' = BM''$. Now, if $M''Q''$ is equal in length in to MQ then the configuration resulting after tracking on AB will be the starting configuration PQM, thereby resulting in a 3-cycle.

The points Q, P', M'' can be obtained by solving the equations

$$x + y = AC, \quad y + z = BC, \quad z + x = AB,$$

where x, y and z denote the distances of Q, P', M'' from A, C, and B respectively.. The solution to the system of equations is

$$x = \frac{AB + AC - BC}{2}; y = \frac{-AB + AC + BC}{2}; z = \frac{AB - AC + BC}{2}.$$

This establishes the result. □

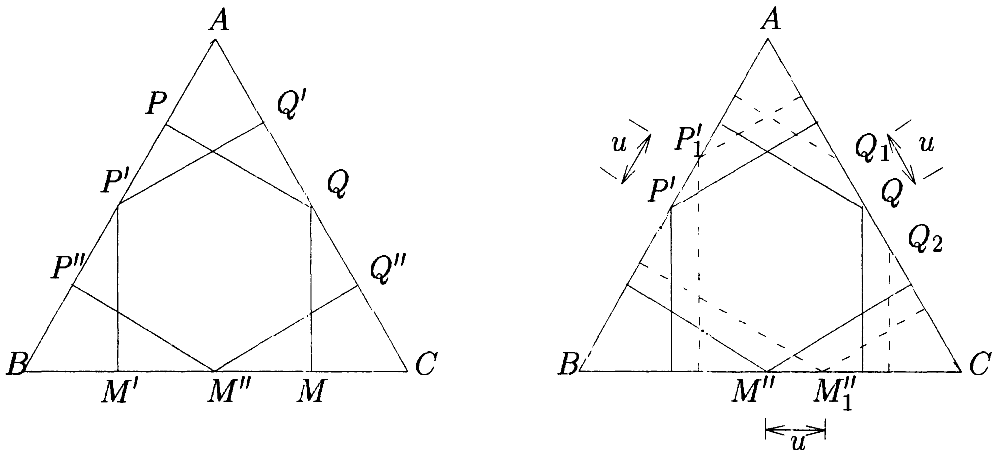

Figure 3.8: Regions of stability

In practice, because of control and round-off errors, it is not reasonable to assume that the robot hand doing the rotation could place its fingers on the exact locations derived as in Theorem 3.1.14. To cope with this problem, we analyze what happens when we perturb the initial positions. Can we still guarantee an infinite sequence of robust rotation steps? By how much can we perturb the initial locations of the fingers and still guarantee an infinite sequence of robust rotation steps?

Theorem 3.1.15 *Every acute triangle can undergo stable arbitrary rotations.*

This is shown by constructing regions of stability on the edges of the triangle. Whenever a grasp lies inside these regions, exact blocking rotations results in a 6-cycle fully contained in the regions. Hence perturbing a grasp by a small amount does not affect the cyclic nature of the process. The following lemma shows how to construct the regions of stability.

Lemma 3.1.16 *Let x, y, and z be as in Theorem 3.1.14, $d = min\{x, y, z\}$, and $u \leq d$. If the finger tracking algorithm starts from a configuration with fingers no further than d from P', Q and M'', exact blocking rotations result in a cycle.*

Proof: Consider Figure 3.8. Let $P_1Q_1M_1$ be any blocked configuration such that $QQ_1 = u$. From Observation 1, it follows that if three blocking rotations are carried out then $u = QQ_1 = P'P'_1 = M''M''_1 = QQ_2$. But the necessary and sufficient condition for carrying out the blocking rotations is $u \leq x,\ u \leq y,\ u \leq z$, which is satisfied by assumption on u. By the same argument, if three more rotations are carried out, the fingers return to their initial position, giving a 6-cycle. □

The robust termination protocol $\lambda(\mathbf{p_0}, \ldots, \mathbf{p_3}, \mathbf{f_0}, \ldots, \mathbf{f_3})$ based on Theorem 3.1.15 works as follows. As long as the initial finger locations are inside the stability belt, each one-step tracking proceeds until a vision system detects no further change in the position of the object. When this condition occurs, the object is blocked and the finger locations are inside the stability belt. A new tracking finger can be designated and the entire process can be repeated.

3.2 Rotating Polygons

Every polygon can be included in a triangle by extending three of its edges and thus it can be rotated according to Algorithm 3.3. The added difficulty is in selecting the including triangle and in keeping the fingers on the polygonal segments. Picking an including triangle relates to selecting a grasping edge set. This choice gives flexibility, but also complexity to the problem of rotating polygons. In this section we develop the algorithm for stable, arbitrary rotations of polygons.

3.2.1 Multi-step Rotations

Figure 3.9 shows the multi-step rotation algorithm for polygons. Algorithm 3.3 has been modified by including the data structures to facilitate grasp transition. This addition is necessary for polygons because unlike in the case of triangles, where the grasping edge set is the same for all the one-step tracking, polygons offer more choices. For polygons, it may be necessary to change the grasping edge set by adding a new edge as the tracking edge. The transition to a new tracking edge is not possible if the robot hand has only three fingers, because at all time, three fingers are needed on the surface of the polygon for the grip. Thus, we need robot hands with more than three fingers. We use robot hands with four fingers. One of the fingers is *free*, in other words not involved in the grasp, for each one-step tracking. When the grasping set is modified, the free finger joins the grasp on the new grasping edge, following which the previously tracking finger leaves the grasp and becomes free.

Consider Algorithm 3.4 given in Figure 3.9. The transition graph S represents all the legal transitions between a one-step tracking to another. The vertices of this graph are 3-tuples of edges. If $(e_i, e_j, e_k) \in V$, then there is a force-closure grasp on edges e_i, e_j, e_k and during the one-step rotation the fixed fingers act on e_i, e_j, while the tracking finger acts on e_k. The vertex set V can be calculated efficiently, as shown in [Ngu88]. Vertex (e_i, e_j, e_k) is connected with an edge to all the vertices of the form (e_j, e_k, e_l). The initialization step picks a first grasping edge set. The **while** loop iterates by doing a one-step finger tracking and then by selecting a new grasp.

The most important aspects of this algorithm are how to select the grasping edges for the next step and how to terminate each one-step. The fingers cannot leave the edges, but are there other factors affecting the termination predicate? With a certain termination predicate, is it possible to have an infinite number of steps, thus being able to accomplish arbitrarily large rotations? As for the

Notation: P is the polygon and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$ the fingers of the hand.

p_0, p_1, p_2, p_3 are the contact points of the respective fingers.

θ is the desired reorientation and r the accomplished rotation.

t is the index of the tracking finger and l the index of the free finger.

$S = (V, E)$ is the transition graph.

$V = \{(e_i, e_j, e_k) | \exists$ force closure grasp on polygonal edges $e_i, e_j, e_k\}$.

$E = \{(v_1, v_2) | v_1, v_2 \in V, v_1 = (e_i, e_j, e_k), v_2 = (e_j, e_k, e_l)\}$.

Input: θ.

Preprocessing: Build S.

Initialization: Let $v \in V$.

Assign $\mathbf{f_0}$ to first(v), $\mathbf{f_1}$ to second(v), $\mathbf{f_2}$ to third(v).

Grasp P using $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$, satisfying the force closure property.

$r = 0; t = 0; n = 0; l = 3$

While $r \neq \theta$ **do**

begin

Track using finger $\mathbf{f_t}$, keeping $\mathbf{f_i}, i \neq t, l$ fixed, and $\mathbf{f_l}$ free,

until the termination predicate $\lambda(p_0, \ldots p_3, \mathbf{f_0}, \ldots, \mathbf{f_3})$ is true.

Update r; $n = n + 1$; $t = l$; $l = (l + 1) \bmod 4$.

v = select-adjacent(v); Assign $\mathbf{f_t}$ to new edge of v.

end

Figure 3.9: Algorithm 3.4 for multi-step finger tracking of polygons

triangular case, we rely on the robust termination condition to guarantee stable arbitrary rotations.

3.2.2 Generality and Robustness for Polygons

For a rotation algorithm by finger tracking to be robust, every one-step must have a robust termination condition. In this section we show how to provide a decision procedure for infinite robust rotations of polygons. For a polygon that can be rotated arbitrarily and robustly, we also show how to calculate stability regions on its edges for the initial grasp. As long as the fingers originate in these regions, a cycle of robust rotation steps is guaranteed. Here we assume that each one-step is ended by the blocking condition. The width of the stability regions determines the amount of error that is allowed in the initial data.

Assume that P is the given polygon with n edges, $e_1, e_2, \ldots, e_n$. The polygon is being manipulated by a robot hand with 4 frictionless point fingers.

We look for n-cycles, which are needed to guarantee arbitrary rotations. The main difficulty in generating a cycle is to ensure that the fingers never slide off their respective edges. Our strategy is to build the robust transition graph $S' = (V', E')$, which is a subgraph of the transition graph S. S' has the following structure:

1. $V' = \{(e_i, e_j, e_k) |$ there exists a robust rotation step on polygonal edges $e_i, e_j, e_k\}$.

2. $E' = \{(v_1, v_2) |$ a robust rotation step on edges in v_1 can be followed by a robust step on edges in v_2. $\}$.

Notice that S' is not S restricted to the vertex set V'. It may or may not be possible to rotate robustly from a blocked configuration on edges (e_i, e_j, e_k) to a blocked configuration on edges (e_j, e_k, e_l). The existence of arbitrary robust rotations is related to the existence of a cycle in S', as given by the following result:

Theorem 3.2.1 *A polygon can be rotated robustly and arbitrarily if and only if its associated robust transition graph S' has a cycle.*

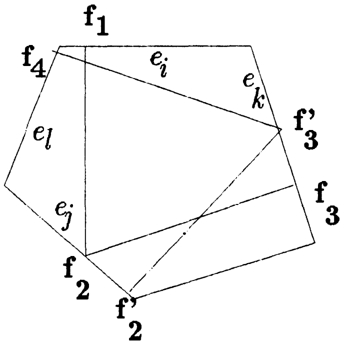

Figure 3.10: A sequence of 2 blocked configurations for a polygon

Thus, a cycle in S' guarantees arbitrary rotations. We now show the procedure for building S'. Consider Figure 3.10. It represents the locations of the fingers on the edges of the polygon for two successive robust rotation steps. The fingers after the first robust rotation step are $\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$. Notice that $(\mathbf{f_1}, \mathbf{f_2}) \perp e_i$ and $(\mathbf{f_2}, \mathbf{f_3}) \perp e_k$. In the next step, $\mathbf{f_2}$ and $\mathbf{f_3}$ are fixed and $\mathbf{f_4}$ pushes. This rotation step is robust if there exist new locations $\mathbf{f'_2}, \mathbf{f'_3}$ for the fixed fingers with $(\mathbf{f'_2}, \mathbf{f'_3}) \perp e_j, (\mathbf{f'_3}, \mathbf{f_4}) \perp e_l$, and the length between $\mathbf{f'_2}$ and $\mathbf{f'_3}$ is the same as the length between $\mathbf{f_2}$ and $\mathbf{f_3}$. A necessary condition to go from one blocked configuration to another is to be able to draw mutual perpendiculars of equal length between e_j and e_k. The first perpendicular is between the the center of the grip and the pushing finger. The second perpendicular is between the two fixed fingers of the next iteration. We use this observation to build a cycle of S' by means of an auxiliary graph denoted by PG, and called *the pairing graph*. $PG = (PGV, PGE)$ is a weighted graph, where:

1. $PGV = \{e_i | e_i \text{ is a polygonal edge }\}$.

2. $PGE = \{(e_i, e_j) |$ there exist mutual perpendiculars of equal length between e_i and $e_j\}$.

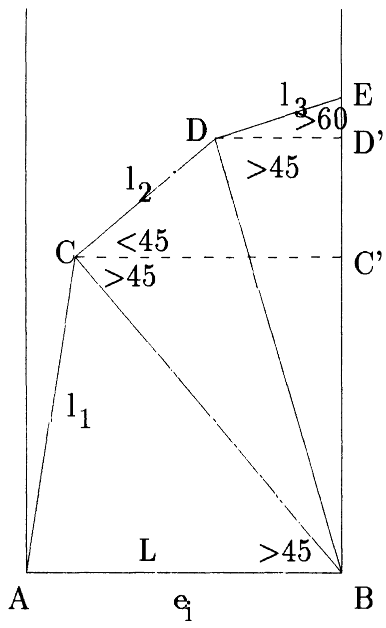

Figure 3.11: An edge and its shadow

3. The weight of each edge is the range of the mutual perpendiculars of equal length.

4. Two *PG*-edges are compatible if their weights overlap.

Theorem 3.2.2 *The graph S' has a cycle if and only if the graph PG has a cycle of compatible edges.*

Definition 3.2.1 *The shadow of an edge is the region delimited by the two lines normal to the edge at its endpoints.*

The following proposition gives an insight into the structure of a pairing graph.

Proposition 3.2.3 *The number of edges of PG is* $O(n)$, *where n is the number of edges of P.*

Proof:

We first show that each edge can be paired with at most six edges of greater or equal length. Let L be the length of the edge e_i to be paired. Every edge that can be paired with e_i must intersect its shadow. Among these edges, at most two are not fully contained in the shadow, one on each side. Let us consider the edges in the shadow, as pictured in Figure 10. Assume for now that there are three edges of length greater than L, adjacent to each other and to e_i, and all of them have positive slopes with respect to $e_i = AB$. Let AC be of length $l_1 > L$, CD be of length $l_2 > L$ and DE of length $l_3 > L$. Since $l_1 > L, \angle(AB, BC) > \frac{\pi}{4}$. Then for the shadow of CD to intersect AB it must be that $\angle(BC, CD) < \frac{\pi}{2}$. If CC' is parallel to AB, which is parallel to DD', then $\angle(C'C, CD) < \frac{\pi}{4}$, thus implying that the length of DD' is smaller that $\frac{L}{2}$. Then for l_3 to be bigger than L it must be that $\angle(D'D, DE) > \frac{\pi}{3}$. Notice that since $DD'$is parallel to $AB, \angle(BD, DD') > \frac{\pi}{4}$. But this means that $\angle(BD, DE) > \frac{7\pi}{12}$, or that the shadow of DE can not hit AB. If the long edges are not chained directly, then it is even harder to fit them in the shadow. Thus, AB could be paired with at most four edges fully contained in the shadow (two that have positive slopes, and two that have negative slopes.) Hence each edge could be paired with at most six edges of greater length.

To complete the proof of the lemma we show that if each edge could paired with a constant number c of edges of greater length, then there are at most cn pairings, where n is the total number of edges. Consider a directed graph where each edge is represented by a vertex. If two edges can be paired, there is an edge directed into the vertex representing the edge of longer length. Resolve ties arbitrarily. The size of the graph is the sum of the out-degrees of all vertices, or cn. □

The consequence of Theorem 3.3 and Proposition 3.4 is that the cycles of robust rotation steps have length at most $O(n)$, and they can be determined in $O(n)$ time.

Let e_i, e_j be two edges, where e_j is contained in the shadow of e_i. The shadow of e_j acts either by spilling to the right of e_i, denoted by r, or by intersecting e_i, denoted by h, or by spilling to the left of e_i, denoted by l. Notice that for the shadow of e_j to spill to the right, it is necessary that it has a positive slope with respect to e_i, and for it to spill to the left, it is necessary that e_j has a negative slope with respect to e_i.

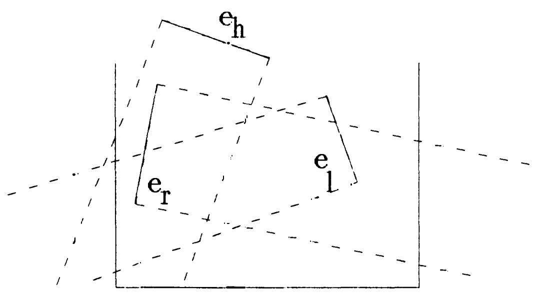

Figure 3.12: The three actions of shadows on an edge

Proposition 3.2.4 *If e is the edge of smallest length in a convex polygon, and we consider the edges in its shadow in a clockwise order, then the only valid sequences for the action of their shadows on e are of the form* $h^*r^*h^*l^*h^*$.

Proof: Let us examine the edges in the shadow in a clockwise order (or equivalently, in decreasing order of their slopes.) If the shadow of edge e_i spills to the left, the shadow of no edge e_j with $j > i$ can spill to the right, by convexity. Then no subsequences of the form lh^*r are allowed, which is described by $(r + h)^*(l + h)^*$. Moreover, if e_i spills to the right and e_{i+1} hits, then $\forall j > i+1, e_j$ can not spill to the right. This is because if v is the projection of v_{i+2} on e, then length$(v1_e, v) >$ length$(v, v2_e)$. Here $v1_e, v2_e$ are the intersection of the shadow of e with the polygon. Therefore, by convexity, no other edge that forms an acute angle can fit in the shadow, so the next edge either hits or spills to the left. This argument shows that no subsequences rh^+r are possible. Similarly, we can show that no subsequences lh^+l are possible. Thus, the only legal sequences are of the form $h^*r^*h^*l^*h^*$. □

Proposition 3.2.5 *If e is the smallest length edge of a convex polygon with n edges, then we can find all the pairings for e in* O(log$n$).

Proof: Let us examine the shadows of the edges that intersect the shadow of e in a clockwise order. By Proposition 3.4, the only possible sequences are of the form $h^*r^*h^*l^*h^*$. Moreover, from the proof of Proposition 3.4 we know that there can be at most three pairings on each side (one edge not fully contained in the shadow, and two edges fully contained in the shadow.) Thus, we can check for these edges separately in the beginning. Then we only need to discover the hits of sequences of the form $r^*h^*l^*$. This can be done in O(log$n$) by binary search. □

Corollary 3.2.6 *The pairing graph of a convex polygon can be built in* O(n log$n$). *The pairing graph of a simple polygon can be built in $O(n^2)$.*

Proof: The following algorithm is O($n$log$n$) for convex polygons:

1. Sort all the edges according to length
2. While the edge list is not empty
   - Let e = edge of smallest length.
   - Find all the pairings of e as shown in Proposition 3.3.
   - Remove e from the sorted list.

For arbitrary polygons, the naive algorithm which check every edge against every other edge has complexity $O(n^2)$ □

Corollary 3.2.7 *A polygon can be rotated robustly and arbitrarily if and only if its associated pairing graph has a cycle of compatible edges.*

To actually find the exact location of the cycle of PG, let $e_0, \ldots, e_{n-1}$ be the edges of the candidate cycle. For each of these edges, let l_i denote the length of the segment obtained by intersecting the line supporting e_i with the line supporting its adjacent edges in the cycle, e_{i-1} and $e_{(i+1)\mathrm{mod}\ n}$. For edge e_i, let x_i and $x_{(i+1)\mathrm{mod}\ n}$ denote the length of the segments determined by a blocked contact point along the extension of e_i determined by its neighbors. A simple geometric insight [Rus90] leads to the following system:

$$
\begin{aligned}
x_0 + x_1 &= l_0 \\
x_1 + x_2 &= l_1 \\
\ldots &= \ldots \\
x_{n-1} + x_0 &= l_{n-1}
\end{aligned}
$$

It follows that $x_j = \frac{1}{2}\sum_{i=0}^{n-2}(-1)^i l_{(j+i)\mathrm{mod}\ n} + l_{(j-1)\mathrm{mod}\ n}$, for $j = 0, \ldots, n-1$. If all these values are positive, then we have an actual cycle. The initial configuration for the grasping fingers that result in this cycle is given by x_0.

Corollary 3.2.8 *Given a desired rotation and a polygon that can be rotated robustly and arbitrarily, the number of one-step trackings can be determined in* O(1) *time. If the polygon can not be rotated infinitely, the maximum robust rotation angle and the number of one-steps can also be computed in* O(1).

We now show how to construct a cycle for S' from a cycle for PG. The vertices of this cycle are obtained by listing, in order, all the adjacent triples of edges participating in the pairing graph cycle. Each S' vertex is connected to its predecessor and successor in the listing. This completes the procedure for finding a cycle that allows for arbitrary rotations with robust termination conditions. To make this algorithm robust in the sense of Definition 3.1, we must show that the cycle is stable. In other words, small imprecisions in the finger positioning must still result in cyclic behavior.

Theorem 3.2.9 *For any polygon that can be rotated robustly for an infinite number of steps, we can calculate start-up regions for the fingers so that when the initial grasp contact lie within those regions, they stay inside the regions for every rotation step.*

Proof: We use the same notation as in Corollary 3.8. To find the stability region around the cycle given by the decision procedure of Corollary 3.8, in other words the length d of the maximum region that is guaranteed to keep the fingers inside the region for an infinite sequence of robust rotation steps, we consider the following system of equations.

$$\begin{aligned} x_0 + x_1 - d &= l_0 \\ x_1 + x_2 - d &= l_1 \\ \ldots &= \ldots \\ x_{n-1} + x_0 - d &= l_{n-1} \end{aligned}$$

In addition, we have that $x_j \geq d$ for $j = 0, \ldots, n-1$. Let

$$m = \min_{j=0}^{n-1}\{\sum_{i=0}^{n-2}(-1)^i l_{(j+i)\bmod n} + l_{(j-1)\bmod n}\}.$$

If follows that $d \leq m$. □

3.3 Summary

In this chapter we have presented a stable algorithm for arbitrary rotations of polygons, and a decision procedure to determine whether a polygon can be rotated robustly and arbitrarily or not. We have done this by introducing the geometric termination condition of blocking in Algorithm 2.1. We then showed how to build the graph that represents the valid sequences of grasps that allow blocking. A cycle of this graph gives the sequence of edges that yields an infinite number of robust rotation steps by finger tracking. For such a sequence of edges, we also showed how to determine start-up regions for the finger tips with the property that when the fingers originate anywhere in the regions and they do robust finger tracking, the resulting behavior is cyclic.

Chapter 4

The Framework

In this chapter we define our notion of manipulation which focuses on the rotational component of the motion. We also propose a framework to formulate, reason about, and solve constraint-based manipulation problems. In Section 4.1 we propose a definition for manipulation. In Section 4.2 we give the formalization of the constraint-based manipulation problem formulation. In Section 4.3 we present an example.

4.1 The Definition of Manipulation

Definition 4.1.1 *Manipulation is the repositioning of a part inside the grip, while maintaining the grip.*

This is a general statement about fact that reorientation is accomplished with respect to the palm of the hand, or more precisely, with respect to a system of coordinates which is fixed on the robot hand. To make Definition 4.1.1 more rigorous, recall that in the process of reorientation, the grasped object undergoes a Euclidean motion, composed of a translation and a rotation. The focus is on measuring rotations, and thus we abstract out the translational component of the motion. The following definitions formalize this idea.

Definition 4.1.2 *Two congruent objects have equivalent orientation if there is a translation that takes one into the other.*

Definition 4.1.2 gives rise to an equivalence relation on the set of object orientations. The equivalence classes are called orientations.

Definition 4.1.3 *Let θ and θ' be orientation classes and object $P \in \theta$. The reorientation of P from θ to θ' is the element of $SO(3)$ that takes P to some element of θ'.*

The most important aspect of dexterous manipulation is being able to guarantee a certain behavior as objects come in contact. Thus, the interaction between the object and the hand is a very important component of manipulation. The role of the geometries of the object and of the manipulator must be clearly defined. The nature of the high-level primitives for doing task-level manipulation is determined by this interaction.

Based on how the hand and the object to be manipulated interact during the actual reorientation phase, we define two classes of algorithms. The first class contains algorithms for which the object is passive inside the grip. In other words, the object is fixed with respect to the finger tips. The fingers perform a rotational motion. Since the object is secured inside the grip, it follows the motion of the fingers. The challenge for this class of algorithms is not in generating the rotational motion, but in coping with the fact that the fingers have limited reachability. Existing trajectory control algorithms can be employed to generate the finger paths for the rotational motion. When any one of the grasping fingers attains its limit of reachability the fingers must be repositioned. The regrasping is done by walking the fingers to their initial grasping regions. We call the algorithms resulting from such an interaction *finger walking algorithms.*

In the second type of relationship, the object is active inside the grip. The internal forces of the grip are used to move the object relative to the grasping fingers. The manipulator produces motion by applying forces along curves on the surface of the object. For this type of algorithms, the challenge is to determine the finger trajectories and forces. We call the manipulation algorithms which result from an active interaction between the fingers and the object *finger tracking algorithms.*

4.2 The Formalization

A dexterous manipulation problem is a system of the form:

$$\mathcal{S} = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{M}),$$

where $\mathcal{B}$ is the set of objects being manipulated by a set of hands $\mathcal{F}$, under the set of constraints $\mathcal{C}$, in a set of spaces $\mathcal{M}$.

$\mathcal{B}$ is called the *body* set. It represents objects with a common geometric property that can be utilized computationally. For instance, the body set could be the set of convex three-dimensional objects with planar faces, or convex two-dimensional objects with quadric surfaces. The geometric properties and insights into classes of objects are powerful in deriving and simplifying the algorithms operating on them. For each instantiation of the body set additional properties of the object are represented, which may include a complete or an approximate geometric description.

$\mathcal{C}$ is called the *constraint* set. It represents the set of constraints for the manipulation problem. All manipulation problems can be expressed in terms of the interaction between the hand and the object. These intereactions are represented equationally as constraints. The weakest constraint set comprises the condition for the robot hand to maintain the grasp on the object being reoriented. This constraint set may be enriched by restricting further the allowed motion. For instance, the constraint set may include equations describing a certain pattern for the interaction between the grasping fingers and the object, such as requesting that one of the fingers stays fixed in space and in continuous contact with the object.

A kinematically enriched constraint set has the advantage of adding structure to the manipulation problem, thus simplifying the control.

$\mathcal{F}$ is called the *hand* set. It represents robot hands with a common structure. This includes the number of fingers of the hand, the structure of each finger, the organization of the fingers, the type of contact between the fingers and the object being manipulated. For instance, $\mathcal{F}$ could be the set of robot hands with three fingers and an opposing thumb, where each finger has the same structure composed of three knuckles, and the finger tips have a point contact with friction on the object. For each instantiation of $\mathcal{F}$, additional properties are represented, which may include the location of the fingers on the object and the forces exerted by each finger.

$\mathcal{M}$ is called the *configuration space.* It represents the space in which the motion of the body can take place. The motion of the object is generally described as a curve in the configuration space. For instance, for a planar object constrained to be in contact with three point fingers in general position, $\mathcal{M}$ is one point, since the motion of the object is fully constrained. The configuration space has great computational importance in defining the control strategy for a desired manipulation.

The various aspects of the manipulation problem can be expressed as functions between the components of $\mathcal{S}$. For a given class of objects and a given set of constraints, what is the algorithm to calculate the configuration space? For a class of objects, a set of constraints, and a path in configuration space, what is the algorithm that results in controlling the object along the trajectory while maintaining the constraints? For task-directed applications where the configuration space is specified, what is the constraint set that yields the simplest control? These are only a few examples of problems that could be formulated in terms of the relationship between the components of $\mathcal{S}$.

The problem of defining the configuration space associated with a class of objects and a given constraint set can be formulated as the function:

$$g : \mathcal{B} \times \mathcal{C} \rightarrow \mathcal{M}.$$

The study of g leads to algorithms to calculate the configuration space. This function captures the manipulability of an object under given constraints, by defining the set of reorientations allowed for the object. $\mathcal{M}$ is naturally related to the group of rigid motions $SE(3)$. Since $SE(3)$ is a topological group with differentiable structure, a curve in $SE(3)$ may be viewed as the trajectory of a rigid body over time. The configuration space $\mathcal{M}$ given by the function g is that subspace of $SE(3)$ which obeys the constraints in $\mathcal{C}$. The paths in $\mathcal{M}$ represent the reorientation trajectories allowed under $\mathcal{C}$. In addition to defining precisely the configuration space, g also gives a qualitative measure of how powerful the constraint space for a given application. If the resulting configuration space is of high dimension, its paths are complex. That is, the object can be manipulated along complicated trajectories. Such trajectories may be an intrinsic part of the application, but others, such as the space-filling curves, may not. If the configuration space has paths that do not belong to the application, the constraint set is too weak. To exclude such trajectories from the configuration space, the constraint set should be enriched. If the dimension of the resulting configuration space is too low, some of the desired reorientations may not be available, implying that the constraint set is too strong and inadequate for the application.

Insights into the structure and properties of $\mathcal{M}$ can lead to better high-level manipulation strategies. In task-level manipulation, the required behavior of the object is often very specific. The structure of the configuration space may be defined by the task. For this type of applications, the challenge is to find the appropriate set of constraints. The constraints may be picked from a predefined set of constraint types. If M is the configuration space defined by the task for a body B, we say that a constraint C' is acceptable if $g(B, C') = M' \supset M$. The constraint set is optimal if M' is the smallest superset of M. The simpler the configuration space, the simpler the control strategies. The problem of finding C'

can be formulated as a function:

$$h : \mathcal{B} \times \mathcal{M} \to \mathcal{C}.$$

Having appropriate definitions for $\mathcal{B}, \mathcal{C}$, and $\mathcal{M}$, the core of the manipulation problem is to generate the control that accomplishes a desired reorientation. What should the robot fingers do? This problem can be formulated as a function relating the body, the constraints, and the curves in the configuration space to the finger motions:

$$f : \mathcal{B} \times \mathcal{C} \times \mathcal{P}(\mathcal{M}) \to \mathcal{F},$$

where $\mathcal{P}(\mathcal{M})$ is the set of paths in the configuration space $\mathcal{M}$.

The function f describes the manner in which the robot uses its fingers to generate a desired trajectory for the object being manipulated. The study of f is essential in addressing various algorithmic questions about the control of dexterous manipulation. It could be used directly to decide what the hand should do in order to generate motion along a given curve in the configuration space. In other words, f shows how to choose the curve in the space of parameters that generates the desired curve in configuration space. Its inverse image $f^{-1}(A) = \{x \in \mathcal{M} | f(x) \in A\}$ could also be used to determine the resultant motion of the body, given a set of forces applied by the hand.

In order to study f, the relationship between the motion of the body and the forces exerted by the hand is needed. This leads to the accelerations along the trajectory of motion. A natural computational tool that supports this kind of applications is provided by Lie groups and Lie algebras. To study velocity and acceleration properties of a trajectory, the natural way is to look at the tangent space of the motion. Since the motion takes place in a subspace of a Lie group, it is convenient to exploit the Lie group structure. The velocity and acceleration constraints of $\mathcal{M}$ can be obtained by moving to the Lie algebra of $SO(3)$.

4.3 Example: Rotating Triangles

To illustrate the ideas of the previous section, we consider a simplified version of the problem described in Chapter 3, by restricting the convex polygons to triangles. The finger tracking control algorithm is derived from the following instantiation of the manipulation system.

1. $\mathcal{B}$ is the set of triangles. Let B be an element of this set.

2. $\mathcal{C}$ is the set of constraints that expresses a continuous three finger grasp on the object. In addition there are two kinematic constraints that describe the fact that at all time, two of the robot fingers stay fixed in the plane. Moreover, the manipulation is being done under the laws of Newtonian mechanics. Let C be an element of this set.

3. $\mathcal{F}$ is the set of robot hands with three fingers. Each finger has a limitless reach region, and the finger tips are frictionless points. Let F be an element of this set.

4. $\mathcal{M}$ is the configuration space defined by these sets. Let M be an element of this set.

The problem is to define the control function that accomplishes the reorientation of the triangle under the desired angular acceleration $\dot{\omega}$.

Each $B \in \mathcal{B}$ is described in terms of its vertices. Let $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}$ be the vectors that correspond to the three vertices. The contact locations of the fingers on the edges of the triangle are also given. Let the vectors $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$ denote these points. The contacts are labeled so that contact $\mathbf{q_i}$ opposes vertex $\mathbf{p_i}$. $\mathbf{q_0}$ and $\mathbf{q_1}$ are the fixed fingers and $\mathbf{q_2}$ is the finger free to move in any way that maintains the grasp. The normal to the edges of the triangle are also known and are labeled by $\boldsymbol{\nu}_0, \nu_1$, and $\boldsymbol{\nu}_2$, where $\boldsymbol{\nu}_i$ is the normal to the edge opposing vertex $\mathbf{p_i}$. The center of mass is at $\mathbf{r_0}$, which is the origin of the system of coordinates.

Each $F \in \mathcal{F}$ is described in terms of the finger contacts $\mathbf{q_0}, \mathbf{q_1}$, and $\mathbf{q_2}$, and in terms of the forces exerted by these fingers $f_0\boldsymbol{\nu}_0, f_1\boldsymbol{\nu}_1$, and $f_2\boldsymbol{\nu}_2$. f_0, f_1, and f_2 are the force magnitudes. Since the contacts are modeled as frictionless points, the forces act in the respective normal directions.

The constraint set $C \in \mathcal{C}$ needs derivations. The requirement that the grasp is maintained at any point in time is expressed by the three equations

$$\det(\overline{\mathbf{q_i}}, \overline{\mathbf{p_j}}, \overline{\mathbf{p_k}}) = 0,$$

appended with the inequalities that constraint the point $\mathbf{q_i}$ on the edge segment determined by $\mathbf{p_j}$ and $\mathbf{p_k}$. Here, (i, j, k) is a permutation of $(0, 1, 2)$. Let $\mathbf{T} \in SE(2)$ be a matrix describing the position of the object relative to a frame of reference fixed on the robot hand. Then $\mathbf{S} = \mathbf{T}^{-1}$ is a matrix describing the position of the robot hand relative to a frame of reference fixed on the object. The three constraints at all time are thus given by $\det(\mathbf{T}\overline{\mathbf{q_i}}, \mathbf{T}\overline{\mathbf{p_j}}, \mathbf{T}\overline{\mathbf{p_k}}) = 0$. By taking into account that $\mathbf{q_0}$ and $\mathbf{q_1}$ stay fixed in the plane, and by expressing the equations in the object-fixed frame, the constraints can be rewritten as $\det(\mathbf{S}\overline{\mathbf{q_0}}, \overline{\mathbf{p_1}}, \overline{\mathbf{p_2}}) = 0$, $\det(\mathbf{S}\overline{\mathbf{q_1}}, \overline{\mathbf{p_2}}, \overline{\mathbf{p_0}}) = 0$, and $\det(\overline{\mathbf{q_2}}, \overline{\mathbf{p_0}}, \overline{\mathbf{p_1}}) = 0$.

In addition to the three equations that describe the grasping condition and the kinematic constraints, the constraint set has the Newtonian force and torque equations. In deriving these equations for this special application, there are several observations. Since the only rotations are about the $\mathbf{z}$ axis, $\boldsymbol{\Omega} = \omega\mathbf{z}$. The moment of inertia $\mathbf{I} = I_{zz}$ is just a scalar. If θ is the angle between $\mathbf{q_i}$ and $\nu_\mathbf{i}$, then $\mathbf{q_i} \times \boldsymbol{\nu}_i = \|\mathbf{q_i}\|\sin\theta_i\mathbf{z}$, since $\mathbf{q_i} \times \boldsymbol{\nu}_i = \det(\mathbf{q_i}, \boldsymbol{\nu}_\mathbf{i})\mathbf{z}$. The system is initially in equilibrium, *i.e.* the motion starts from rest. The following five equations comprise the constraint set C:

$$\dot{w}I = \sum_{i=0}^{2} f_i \mathbf{q_i} \times \nu_\mathbf{i} \tag{4.1}$$

$$-m\ddot{\mathbf{u}} = \sum_{i=0}^{2} f_i \nu_\mathbf{i} \tag{4.2}$$

$$\det(\mathbf{S}\overline{\mathbf{q_0}}, \overline{\mathbf{p_1}}, \overline{\mathbf{p_2}}) = 0 \tag{4.3}$$

$$\det(\mathbf{S}\overline{\mathbf{q_1}}, \overline{\mathbf{p_0}}, \overline{\mathbf{p_2}}) = 0 \tag{4.4}$$

$$\det(\overline{\mathbf{q_2}}, \overline{\mathbf{p_0}}, \overline{\mathbf{p_1}}) = 0 \tag{4.5}$$

4.3.1 Defining the Configuration Space

The next step in the derivation of the system describing the give problem is to establish the configuration space.

Proposition 4.3.1 *The configuration space for the motion of the constrained triangle is given by*

$$M = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{u}(\mathbf{R}) \\ 0 & 1 \end{pmatrix} \;\middle|\; \mathbf{u}(\mathbf{R}) = -\frac{det(\mathbf{R}\mathbf{q_1}, \mathbf{q_1})}{det(\mathbf{q_0}, \mathbf{q_1})}\mathbf{q_1} + \frac{det(\mathbf{R}\mathbf{q_0}, \mathbf{q_0})}{det(\mathbf{q_0}, \mathbf{q_1})}\mathbf{q_0} \right\}$$

where $\mathbf{R} \in SO(2)$.

Proof: Since the triangle undergoes a Euclidean motion between any two points in time, there is a transformation matrix $\mathbf{T}$, whose inverse is $\mathbf{S}$, which captures this motion:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{u} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & a \\ -\sin\theta & \cos\theta & b \\ 0 & 0 & 1 \end{pmatrix}$$

Here, θ is the angle of rotation and $\overline{\mathbf{u}} = (a, b, 1)^T$ is its corresponding translation.

The kinematic constraints can be phrased by saying that $\mathbf{q_0}$ is a linear combination of $\mathbf{p_1}$ and $\mathbf{p_2}$ and $\mathbf{q_1}$ is a linear combination of $\mathbf{p_0}$ and $\mathbf{p_2}$, in other words, $det(\overline{\mathbf{q_1}}, \overline{\mathbf{p_2}}, \overline{\mathbf{p_0}}) = 0$ and $det(\overline{\mathbf{q_0}}, \overline{\mathbf{p_2}}, \overline{\mathbf{p_1}}) = 0$. Then the constraint that $\mathbf{q_0}$ and $\mathbf{q_1}$ are fixed in space and slide on the sides of the triangle becomes: $det(\overline{\mathbf{q_1}}, \mathbf{T}\overline{\mathbf{p_2}}, \mathbf{T}\overline{\mathbf{p_0}}) = 0$ and $det(\overline{\mathbf{q_0}}, \mathbf{T}\overline{\mathbf{p_2}}, \mathbf{T}\overline{\mathbf{p_1}}) = 0$ or, by multiplying by $\mathbf{S} = \mathbf{T}^{-1}$, $det(\mathbf{S}\overline{\mathbf{q_1}}, \overline{\mathbf{p_2}}, \overline{\mathbf{p_0}}) = 0$ and $det(\mathbf{S}\overline{\mathbf{q_0}}, \overline{\mathbf{p_2}}, \overline{\mathbf{p_1}}) = 0$

By using block notation, observe that $\mathbf{S}\overline{\mathbf{q_i}} = \mathbf{R}_\theta \mathbf{q_i} + \mathbf{u}$. It follows further that $det(\mathbf{R}_\theta \mathbf{q_1}, \mathbf{p_0} - \mathbf{p_2}) + det(\mathbf{u}, \mathbf{p_0} - \mathbf{p_2}) = 0$ and $det(\mathbf{R}_\theta \mathbf{q_0}, \mathbf{p_1} - \mathbf{p_2}) + det(\mathbf{u}, \mathbf{p_1} - \mathbf{p_2}) = 0$ Since $\mathbf{q_0}$ and $\mathbf{q_1}$ are linearly independent, we can express $\mathbf{u} = \alpha \mathbf{q_0} + \beta \mathbf{q_1}$. Then, $det(\mathbf{u}, \mathbf{p_1} - \mathbf{p_2}) = det(\alpha \mathbf{q_0} + \beta \mathbf{q_1}, \mathbf{p_1} - \mathbf{p_2}) = \beta det(\mathbf{q_1}, \mathbf{p_1} - \mathbf{p_2})$ and similarly, $det(\mathbf{u}, \mathbf{p_0} - \mathbf{p_2}) = \alpha det(\mathbf{q_0}, \mathbf{p_0} - \mathbf{p_2})$, or $\alpha = -\frac{det(\mathbf{R}_\theta \mathbf{q_1}, \mathbf{p_0} - \mathbf{p_2})}{det(\mathbf{q_0}, \mathbf{p_0} - \mathbf{p_2})}$ and $\beta = -\frac{det(\mathbf{R}_\theta \mathbf{q_0}, \mathbf{p_1} - \mathbf{p_2})}{det(\mathbf{q_1}, \mathbf{p_1} - \mathbf{p_2})}$. So for each rotation angle θ, the corresponding translation is:

$$\mathbf{u} = -\frac{det(\mathbf{R}_\theta \mathbf{q_1}, \mathbf{q_1})}{det(\mathbf{q_0}, \mathbf{q_1})}\mathbf{q_1} + \frac{det(\mathbf{R}_\theta \mathbf{q_0}, \mathbf{q_0})}{det(\mathbf{q_0}, \mathbf{q_1})}\mathbf{q_0}$$

The system of coordinates can be chosen so that $\mathbf{q_0}$ and $\mathbf{q_1}$ are not parallel the function is well defined. □

The function $u(R)$ that defines the dependency of the translational component of motion to its rotational component in the definition of M encodes the structure of the configuration space given by $g : \mathcal{B} \times \mathcal{C} \to \mathcal{M}$.

The g function is a tool for the study of various kinematic properties of the system. An example is an algebraic formulation of the result that describes the motion of the instantaneous center of rotation.

Proposition 4.3.2 *The position of the instantaneous center of rotation of the triangle is diametrically opposed to the A vertex of the triangle.*

Proof: The same notational conventions as in Proposition 4.3.1 apply. The rotations are clockwise rotation. The instantaneous center of rotation is the point with zero velocity. Let $\mathbf{S}$ be a transformation matrix as in Proposition 4.3.1. Since $\mathbf{R}_\theta{}^{-1}\dot{\mathbf{R}}_\theta = \dot{\theta}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \dot{\theta}\mathbf{J}$, where $\mathbf{J} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. It follows that the derivative of $\mathbf{S}$ is $\dot{\mathbf{S}} = \begin{pmatrix} \dot{\mathbf{R}}_\theta & \dot{\mathbf{u}} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \dot{\theta}\mathbf{R}_\theta \mathbf{J} & \dot{\mathbf{u}} \\ 0 & 0 \end{pmatrix}$.

If $\mathbf{c}$ is the vector denoting the instantaneous center of rotation, then $\dot{\mathbf{S}}\overline{\mathbf{c}} = 0$, or $\dot{\theta}\mathbf{R}_\theta \mathbf{J}\mathbf{c} + \dot{\mathbf{u}} = 0$, or $\dot{\mathbf{u}}/\dot{\theta} = -\mathbf{R}_\theta \mathbf{J}\mathbf{c}$. In other words, $\dot{\mathbf{S}}$ has encoded in it the instantaneous center of rotation. Therefore, the motion constraints are $det(\mathbf{k}, \mathbf{S}\overline{\mathbf{q_0}}, \overline{\mathbf{q_0}}) = 0$ and $det(\mathbf{k}, \mathbf{S}\overline{\mathbf{q_1}}, \overline{\mathbf{q_1}}) = 0$, and by differentiating the motion constraints we obtain the condition on velocity $det(\mathbf{k}, \dot{\mathbf{S}}\overline{\mathbf{q_0}}, \overline{\mathbf{q_0}}) = 0$ and $det(\mathbf{k}, \dot{\mathbf{S}}\overline{\mathbf{q_1}}, \overline{\mathbf{q_1}}) = 0$. Expanding by minors, we get $det(\dot{\theta}\mathbf{J}\mathbf{q_1} + \dot{\mathbf{u}}, \mathbf{q_1}) = 0$ and $det(\dot{\theta}\mathbf{J}\mathbf{q_0} + \dot{\mathbf{u}}, \mathbf{q_0}) = 0$. By linearity, with $\|\mathbf{x}\|$ denoting the norm of $\mathbf{x}$, we have $\dot{\theta} det(\mathbf{J}\mathbf{q_1}, \mathbf{q_1}) + det(\dot{\mathbf{u}}, \mathbf{q_1}) = 0$, or $det(\dot{\mathbf{u}}, \mathbf{q_1}) = -\dot{\theta}\|\mathbf{q_1}\|^2$, and $\dot{\theta} det(\mathbf{J}\mathbf{q_0}, \mathbf{q_0}) + det(\dot{\mathbf{u}}, \mathbf{q_0}) = 0$, or $det(\dot{\mathbf{u}}, \mathbf{q_0}) = -\dot{\theta}\|\mathbf{q_0}\|^2$. Since $det(\mathbf{J}) = 1$ and $\mathbf{J}^2 = -\mathbf{I}$, this is equivalent to $det(\mathbf{J}\dot{\mathbf{u}}/\dot{\theta}, \mathbf{J}\mathbf{q_0}) = -\|\mathbf{q_0}\|^2$ and $det(\mathbf{J}\dot{\mathbf{u}}/\dot{\theta}, \mathbf{J}\mathbf{q_1}) = -\|\mathbf{q_1}\|^2$. At time 0, $\mathbf{R}_\theta = \mathbf{I}$ and so $\mathbf{J}\dot{\mathbf{u}}/\dot{\theta}$ is the center of rotation, $\mathbf{c}$, as observed above. Now because $\mathbf{q_0}$ and $\mathbf{q_1}$ are linearly independent and $\mathbf{J}\mathbf{q_i} = \mathbf{q_i}^{\perp}$, $\mathbf{c} = \gamma(\mathbf{J}\mathbf{q_0}) + \delta(\mathbf{J}\mathbf{q_1})$, and by applying the appropriate linearity properties, we get that $\delta det(\mathbf{J}\mathbf{q_1}, \mathbf{J}\mathbf{q_0}) = -\|\mathbf{q_0}\|^2$ and $\gamma det(\mathbf{J}\mathbf{q_0}, \mathbf{J}\mathbf{q_1}) = -\|\mathbf{q_1}\|^2$, or $\mathbf{c} = \frac{\mathbf{J}}{det(\mathbf{q_1}, \mathbf{q_0})}(\|\mathbf{q_1}\|^2\mathbf{q_0} - \|\mathbf{q_0}\|^2\mathbf{q_1})$. Since $\mathbf{p_2}$ was chosen to be the origin of the system of coordinates, we can conclude that:

$$\mathbf{c} - \mathbf{p_2} = \mathbf{S}\frac{\mathbf{J}}{det(\mathbf{q_1}, \mathbf{q_0})}(\|\mathbf{q_1}\|^2\mathbf{q_0} - \|\mathbf{q_0}\|^2\mathbf{q_1}),$$

which gives the closed form equation for the motion of the instantaneous center of rotation.

This formula is consistent and provides a further quantification of the result in Proposition 3.1.4. The length of $\mathbf{c}$ is given by:

$$\|\mathbf{c}\|^2 = \frac{\|\|\mathbf{q_1}\|^2\mathbf{q_0} - \|\mathbf{q_0}\|^2\mathbf{q_1}\|^2}{\|\mathbf{q_0}\|^2\|\mathbf{q_1}\|^2\sin^2 A} = \frac{\|\mathbf{q_0} - \mathbf{q_1}\|^2}{\sin^2 A}$$

A simple trigonometric calculation involving the law of cosines gives that $\|\mathbf{q_0} - \mathbf{q_1}\| = 2r\sin A$, where r is the radius of the circle determined by A, P, G, Q, and so $|\mathbf{c}| = 2r$, which is constant. The distance from A to R_c is $2r$, hence the instantaneous center of rotation moves so that it diametrically opposes A. $\square$

4.3.2 Defining the Control

The goal of this problem is to derive the control algorithm that commands finger motions to accomplish the desired reorientation. To derive the control strategy for the reorientation according to the given $\dot{\omega}$, we derive the system that yields the force value f_0, f_1, f_2 and the location of q_2. For this, the finger forces should be related to motion, which leads to expressing the kinematic constraints of the problem as acceleration constraints.

Consider the equation $\det(\mathbf{S}\overline{\mathbf{q_i}}, \overline{\mathbf{p_j}}, \overline{\mathbf{p_k}}) = 0$. This is equivalent to

$$\det(\mathbf{R}\mathbf{q_i} + \mathbf{u}, \mathbf{p_j}, \mathbf{p_k}) = 0.$$

The corresponding acceleration constraint is

$$\det(\ddot{\mathbf{R}}\mathbf{q_i} + \ddot{\mathbf{u}}, \mathbf{p_j}, \mathbf{p_k}) = 0, \text{ or}$$

$$\det(\dot{\mathbf{\Omega}} \times \mathbf{q_i} + \ddot{\mathbf{u}}, \mathbf{p_j}, \mathbf{p_k}) = 0.$$

For planar rotations, $\dot{\Omega} = \dot{\omega}\mathbf{z}$ and thus $\dot{\mathbf{\Omega}} \times \mathbf{q_i} = \dot{\omega}\mathbf{q_i}^{\perp}$. This observation can be used to simplify the constraint equations (4.3) and (4.4) to:

$$\det(\dot{\omega}\mathbf{q_0}^{\perp} + \ddot{\mathbf{u}}, \mathbf{p_1} - \mathbf{p_2}) = 0 \tag{4.6}$$

$$\det(\dot{\omega}\mathbf{q_1}^{\perp} + \ddot{\mathbf{u}}, \mathbf{p_0} - \mathbf{p_2}) = 0 \tag{4.7}$$

Equation (2.5) can be simplified further by expressing $\mathbf{q_2}$ in terms of its barycentric coordinates. If s is the barycentric coordinate of $\mathbf{q_2}$, the condition that keeps $\mathbf{q_2}$ on the edge defined by $\mathbf{p_0}$ and $\mathbf{p_1}$ is given by

$$\mathbf{q_2} = s(\mathbf{p_0} - \mathbf{p_1}) + \mathbf{p_1}. \tag{4.8}$$

In other words, the dynamics of the problem is described by the equations (4.1), (4.2), (4.6), (4.7), and (4.8). These equations give a system whose unknows are f_0, f_1, f_2, and s.

We continue by showing how to process the system of equations to solve for the unknowns.

Substituting the values for $\dot{\omega}$ and $\ddot{\mathbf{u}}$ from equations (2.1) and (2.2) into equations (2.6) and (2.7) we get:

$$\sum_{i=0}^{2} f_i \left(\frac{1}{I}\det(\mathbf{q_0}, \boldsymbol{\nu}_1)\mathbf{q_i} \times \boldsymbol{\nu}_i + \frac{1}{m}\langle \boldsymbol{\nu}_i, \boldsymbol{\nu}_1\rangle\right) = 0. \tag{4.9}$$

$$\sum_{i=0}^{2} f_i \left(\frac{1}{I}\det(\mathbf{q_1}, \boldsymbol{\nu}_0)\mathbf{q_i} \times \boldsymbol{\nu}_i + \frac{1}{m}\langle \boldsymbol{\nu}_i, \nu_0\rangle\right) = 0. \tag{4.10}$$

A block form expression for these two equation can be gotten by denoting:

$$\mathbf{A} = \begin{pmatrix} \det(\mathbf{q_0}, \boldsymbol{\nu}_1) \\ \det(\mathbf{q_1}, \boldsymbol{\nu}_0) \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{q_0} \times \boldsymbol{\nu}_0 \\ \mathbf{q_1} \times \boldsymbol{\nu}_1 \end{pmatrix}^T, \mathbf{C} = \begin{pmatrix} (\mathbf{p_0} - \mathbf{p_1}) \times \boldsymbol{\nu}_2 \\ \mathbf{p_1} \times \boldsymbol{\nu}_2 \end{pmatrix}^T,$$

$$\mathbf{N} = \begin{pmatrix} \langle \boldsymbol{\nu}_0, \boldsymbol{\nu}_1\rangle & \langle \boldsymbol{\nu}_1, \boldsymbol{\nu}_1\rangle \\ \langle \boldsymbol{\nu}_0, \boldsymbol{\nu}_0\rangle & \langle \boldsymbol{\nu}_1, \boldsymbol{\nu}_0\rangle \end{pmatrix}, \mathbf{G} = \begin{pmatrix} 0 & \langle \boldsymbol{\nu}_2, \boldsymbol{\nu}_1\rangle \\ 0 & \langle \boldsymbol{\nu}_2, \boldsymbol{\nu}_0\rangle \end{pmatrix}, \mathbf{x} = \begin{pmatrix} f_0 \\ f_1 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} sf_2 \\ f_2 \end{pmatrix}.$$

This notation can be used to express the new constraint equation, which combines with the torque equation, to give the motion system:

$$\left[\frac{1}{I}\mathbf{ab} - \frac{1}{m}\mathbf{N}\right]\mathbf{x} + \left[\frac{1}{I}\mathbf{ac} - \frac{1}{m}\mathbf{G}\right]\mathbf{y} = 0. \tag{4.11}$$

$$\mathbf{I}\dot{\omega} = \mathbf{bx} + \mathbf{cy}. \tag{4.12}$$

The first notable thing about his system is that there are four variables, but only three equations. If no determinant vanishes, this results in a one-parameter family of solutions for any angular acceleration $\dot{\omega}$.

Some algebraic manipulation results in the following equivalent system:

$$x = \mathbf{N}^{-1}m\dot{\omega}\mathbf{a} - \mathbf{N}^{-1}\mathbf{G}\mathbf{y}. \tag{4.13}$$

$$(\mathbf{bN}^{-1}\mathbf{G} - \mathbf{c})\mathbf{y} = \mathbf{I}\dot{\omega} - \mathbf{bN}^{-1}m\dot{\omega}\mathbf{a}. \tag{4.14}$$

Equation (2.13) gives the dependency of the fixed finger forces f_0 and f_1 on the force f_2 and the location s. Equation (2.14) gives a scalar relationship between the two unknowns $f_2 s$ and f_2. This equation can be expressed as $\alpha_0 f_2 + \alpha_1 f_2 s = \alpha_2$, where α_0, α_1, and α_2 are constant terms obtained by doing the multiplications in (2.14), which gives the physical interpretation of why there should be a one-parameter family of solutions, rather than a unique solution. Given $\dot{\omega}$, finger f_2 can be placed anywhere on the edge determined by $\mathbf{p_0}$ and $\mathbf{p_1}$, and its pushing force is dependent on this location: $f_2 = \frac{\alpha_2}{\alpha_0+\alpha_1 s}$. From this dependency, it follows that the system has a singularity when $s = -\frac{\alpha_0}{\alpha_1}$. This singularity corresponds to the location s for which the three fingers form a force-direction closure grip. It is intuitively clear that the triangle does not move when in a force-closure grip, no matter how large the forces become.

4.4 Summary

In this chapter we have proposed a framework for describing manipulation planning problems. A manipulation system can be expressed algebraically in terms of the description of the class of objects being manipulated, the class of hands performing the manipulation, and the constraints for the motion. A manipulation plan is described as a function between these sets. The structure of the manipulation system, the function defining the configuration space, and the function defining the control have been demonstrated for the example of triangles being manipulated by robot hands with three frictionless contacts.

Chapter 5

Manipulation in Three Dimensions

In this chapter we present an algorithm for rotating polyhedra by finger tracking. This algorithm results from a formulation of the problem in the framework of Chapter 4. Its strategy allows for arbitrary rotational motions for the grasped object by finger tracking. For task-directed manipulation algorithms where the motion of the object can be expressed as a sequence of simple time-varying functions, the algorithm for finger tracking can be used as a primitive. Of particular importance is the class of rotational motions whose axes remain constant over time. These are the planar rotational motions. This type of rotation is important in itself, since many fine assembly procedures require the planar reorientation of a part, for the purpose of alignment. Moreover, any overall rotation can be obtained by composing such planar reorientations. Our algorithm provides for a simple solution to control the planar rotational motions of a grasped object. This is an important primitive for high-level, task-directed, fine manipulation which can isolate the robot programmer from the complexity of low-level manipulation control.

5.1 Rotations of Polyhedra

To formulate the problem of rotating polyhedra, consider the following instantiation of the manipulation system $\mathcal{S}$ defined in Chapter 4.

1. $\mathcal{B}$ is the set of polyhedra. Let B be an element of this set.

2. $\mathcal{C}$ is the set of constraints that expresses a continuous four finger grasp on the object. In addition there are three kinematic constraints that describe the fact that at all time, three of the robot fingers stay fixed in space. Moreover, the manipulation is being done under the laws of Newtonian mechanics. Let C be an element of this set.

3. $\mathcal{F}$ is the set of robot hands with four fingers. Each finger has a limitless reach region, and the finger tips are frictionless points. Let F be an element of this set.

4. $\mathcal{M}$ is the configuration space defined by these sets. Let M be an element of this set.

The problem is to define the control function that accomplishes the reorientation of the polyhedron under the desired angular acceleration $\dot{\mathbf{\Omega}}$. For simplicity of exposition, we show the derivation when B is a tetrahedron. The extension to polyhedra is given in Section 5.1.4.

Each $B \in \mathcal{B}$ is described in terms of its vertices. Let $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ be the vectors that correspond to the four vertices. In addition, let $\mathbf{p_i'} = \mathbf{p_i} - \mathbf{p_3}$ for $i = 0, 1, 2$. The contact locations of the fingers on the edges of the tetrahedron are also given. The vectors $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$ denote these points. The indices are chosen so that $\mathbf{q_i}$ opposes vertex $\mathbf{p_i}$. The contacts $\mathbf{q_0}, \mathbf{q_1}$, and $\mathbf{q_2}$ correspond to the fixed fingers and $\mathbf{q_3}$ is the finger free to track in a way that maintains the grasp. The normal to the edges of the tetrahedron are also known and are labeled by $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2$, and $\boldsymbol{\nu}_3$, where $\boldsymbol{\nu}_i$ is the normal to the edge opposing vertex $\mathbf{p_i}$. The center of mass is at $\mathbf{r_0}$, which is the origin of the system of coordinates.

Each $F \in \mathcal{F}$ is described in terms of the finger contacts $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$, and $\mathbf{q_3}$, and in terms of the forces exerted by these fingers $f_0\boldsymbol{\nu}_0$, $f_1\boldsymbol{\nu}_1$, $f_2\boldsymbol{\nu}_2$, and $f_3\boldsymbol{\nu}_3$, where f_0, f_1, f_2, and f_3 are the force magnitudes. Since the contacts are modeled as frictionless points, the forces act in the respective normal directions. An algorithm based on frictionless contacts may not be entirely general, but nonetheless, it has interesting features. In some application domains that are characterized by low coefficients of friction, such as the manipulation under water or the cleaning of a spill, the frictional forces can be ignored. However, the extension to hardfinger contacts is shown in a later section.

The constraint set $C \in \mathcal{C}$ needs derivations. The tetrahedron is constrained to be grasped continuously by the fixed fingers at $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_3}$ and the tracking finger at $\mathbf{q_3}$. In addition, the manipulation operation is performed under the laws of Newtonian mechanics.

5.1.1 Defining the Constraint Equations

The constraint set comprises the equations that maintain the grasp, the equations that maintain the three kinematic constraints, and the Newtonian force and torque equations governing the motion. The nature of these equations is determined relative to a reference frame. The natural choices are to fix the reference frame on the robot hand, or to fix the reference frame on the object. The first alternative provides an inertial system, in which the constraints that keep contacts $\mathbf{q}_i$ on the face opposite vertex $\mathbf{p}_i$ can be expressed by the equation that states the coplanarity of four points. With our hypothesis, and with a reference frame fixed on the robot hand, this constraint can be described by the equations $\det(\overline{\mathbf{q_i}}, \mathbf{T}\overline{\mathbf{p_j}}, \mathbf{T}\overline{\mathbf{p_k}}, \mathbf{T}\overline{\mathbf{p_l}}) = 0$, where (i, j, k) is an even permutation of $(0, 1, 2)$. (These equations merely put the contacts on the correct plane. The contacts can be forced to stay within the interiors of a face with additional inequality constraints.) By fixing the reference frame on the object, the constraint equations can be expressed in a similar fashion. The equations are $\det(\mathbf{S}\overline{\mathbf{q_i}}, \overline{\mathbf{p_j}}, \overline{\mathbf{p_k}}, \overline{\mathbf{p_l}}) = 0$, for (i, j, k) even permutations of $(0, 1, 2)$. Although the reference frame fixed on the object is no longer inertial because during the manipulation step the body accelerates, we prefer this choice. There are several reasons for our selection. It can be seen clearly that the constraint equations are much simpler from a computational point of view. In addition, in the control phase of the problem we are faced with monitoring the way in which the various features of the object move. A reference frame fixed on the hand follows the motion of th body in configuration space. The configuration space has a complicated structure even for the simple case of polyhedra, as we will see in the next section. In [Rus91] we have shown that the vertices of a grasped tetrahedron subject to the given constraints may move along space-filling curves. A reference frame on the body follows the motion of the contact points slide relative to the faces of the object. Since the faces of polyhedra are given by very simple equations, this is a clear advantage. The main disadvantage for this choice has to do with the fact that such a system is no longer inertial. Thus, the usual Newtonian force and torque equations have to be modified for our system.

The requirement that the grasp is maintained at any point in time is expressed by the three equations $\det(\mathbf{q_i}, \mathbf{p_j}, \mathbf{p_k}, \mathbf{p_l}) = \mathbf{0}$ appended with the inequalities that constraint the point $\mathbf{q_i}$ on the face determined by $\mathbf{p_j}, \mathbf{p_k}$, and $\mathbf{p_l}$. Here, (i, j, k, l) is a permutation of $(0, 1, 2, 3)$. To derive the equations for the kinematic constraints, let $\mathbf{T} \in SE(3)$ be a matrix describing the position of the object relative to a frame of reference fixed on the robot hand. Then $\mathbf{S} = \mathbf{T}^{-1}$ is a matrix describing the position of the robot hand relative to a frame of reference fixed on the object. The three constraints that keep contacts $\mathbf{q_i}$, $i = 0, 1, 2$ fixed in space are thus given by $\det(\mathbf{Tq_i}, \mathbf{Tp_j}, \mathbf{Tp_k}, \mathbf{Tp_l}) = 0$. By taking into account that $\mathbf{q_0}, \mathbf{q_1}$ and $\mathbf{q_2}$ stay fixed in the plane, and by expressing the equations in the object-fixed frame, the constraints can be rewritten as $\det(\mathbf{S}\mathbf{q_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}) = 0$, $\det(\mathbf{S}\mathbf{q_1}, \mathbf{p_2}, \mathbf{p_0}, \mathbf{p_3}) = 0$, $\det(\mathbf{S}\mathbf{q_2}, \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_3}) = 0$, and $\det(\mathbf{q_3}, \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}) = 0$.

Proposition 5.1.1 *The Newtonian force equation valid for the object-fixed system of coordinates is:*

$$(\mathbf{\Omega}_t \times (\mathbf{\Omega}_t \times (\mathbf{r} - \mathbf{u}_t)) - \dot{\mathbf{\Omega}}_t \times (\mathbf{r} - \mathbf{u}_t) + 2\mathbf{\Omega}_t \times \dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t = \frac{1}{m} \sum f_i \boldsymbol{\nu}_i. \tag{5.1}$$

Proof: In object-fixed frame, $\mathbf{r}_0$ is the position of the center of mass at all time. Then $\mathbf{r}_t = \mathbf{T}_t\mathbf{r}_0 = \mathbf{S}_t^{-1}\mathbf{r}_0 = \mathbf{R}_t^{-1}\mathbf{r}_0 - \mathbf{R}_t^{-1}\mathbf{u}_t$. Let $\mathbf{P}_t = \mathbf{R}_t^{-1}$. We have that $\mathbf{r}_t = \mathbf{P}_t\mathbf{r}_0 - \mathbf{P}_t\mathbf{u}_t$, and by differentiation, $\dot{\mathbf{r}}_t = \dot{\mathbf{P}}_t(\mathbf{r}_0 - \mathbf{u}_t) - \mathbf{P}_t\dot{\mathbf{u}}_t$. By further differentiation, $\ddot{\mathbf{r}}_t = \ddot{\mathbf{P}}_t(\mathbf{r}_0 - \mathbf{u}_t) - 2\dot{\mathbf{P}}_t\dot{\mathbf{u}}_t - \mathbf{P}_t\ddot{\mathbf{u}}_t$. For simplicity, we express these equations in terms of Lie algebra calculations. By definition, we have that $\mathbf{P}_t\mathbf{R}_t = \mathbf{I}$, which implies that $\dot{\mathbf{P}}_t\mathbf{R}_t + \mathbf{P}_t\dot{\mathbf{R}}_t = 0$. That is, $\dot{\mathbf{P}}_t = -\mathbf{P}_t\dot{\mathbf{R}}_t\mathbf{P}_t$ and thus $\ddot{\mathbf{P}}_t = 2\mathbf{P}_t\dot{\mathbf{R}}_t\mathbf{P}_t\dot{\mathbf{R}}_t\mathbf{P}_t - \mathbf{P}_t\ddot{\mathbf{R}}_t\mathbf{P}_t$. In other words, we have shown that :

$$\begin{aligned}
\mathbf{P}_t^{-1}\dot{\mathbf{P}}_t &= -\mathbf{P}_t^{-1}\mathbf{P}_t\dot{\mathbf{R}}_t\mathbf{P}_t = -\dot{\mathbf{R}}_t\mathbf{R}_t^{-1} = -\mathbf{Y}_t, \\
\mathbf{P}_t^{-1}\ddot{\mathbf{P}}_t &= 2(\dot{\mathbf{R}}_t\mathbf{R}_t^{-1})^2 - \ddot{\mathbf{R}}_t\mathbf{R}_t^{-1} = 2\mathbf{Y}_t^2 - \dot{\mathbf{Y}}_t - \mathbf{Y}_t^2 = \mathbf{Y}_t^2 - \dot{\mathbf{Y}}_t.
\end{aligned}$$

By substituting these values in the velocity and acceleration equations, we have that $\mathbf{P}_t^{-1}\dot{\mathbf{r}}_t = -\mathbf{Y}_t(\mathbf{r}_0 - \mathbf{u}_t) - \dot{\mathbf{u}}_t$, and $\mathbf{P}_t^{-1}\ddot{\mathbf{r}}_t = (\mathbf{Y}_t^2 - \dot{\mathbf{Y}}_t)(\mathbf{r}_0 - \mathbf{u}_t) + 2\mathbf{Y}_t\dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t$. Let us go back to the force equation. Since there is no friction, all forces are normal, and we have $\ddot{\mathbf{r}}_t = \frac{1}{m}\sum_{i=0}^{3} f_i\boldsymbol{\nu}_i$. We have shown the transformation for $\ddot{\mathbf{r}}_t$. We need to process further the normals, because in space-fixed coordinates, they change their direction. At time t, $\boldsymbol{\nu}_0$ is the normal to face 0 which is spanned by $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$:

$$\boldsymbol{\nu}_0 = \frac{\mathbf{T}_t(\mathbf{p_1}-\mathbf{p_3})\times\mathbf{T}_t(\mathbf{p_2}-\mathbf{p_3})}{\|\mathbf{T}_t(\mathbf{p_1}-\mathbf{p_3})\times\mathbf{T}_t(\mathbf{p_2}-\mathbf{p_3})\|} = \frac{\mathbf{P}_t(\mathbf{p_1}-\mathbf{p_3})\times\mathbf{P}_t(\mathbf{p_2}-\mathbf{p_3})}{\|\mathbf{P}_t(\mathbf{p_1}-\mathbf{p_3})\times\mathbf{P}_t(\mathbf{p_2}-\mathbf{p_3})\|}.$$

Since $\mathbf{P}_t$ is a rotation, $\mathbf{P}_t(\mathbf{p_1} - \mathbf{p_3}) \times \mathbf{P}_t(\mathbf{p_2} - \mathbf{p_3}) = \mathbf{P}_t((\mathbf{p_1} - \mathbf{p_3}) \times (\mathbf{p_2} - \mathbf{p_3}))$. By using the invariance of the 2-norm under orthogonal transformations, it follows that

$$\boldsymbol{\nu}_0 = \mathbf{P}_t\frac{(\mathbf{p_1}-\mathbf{p_3})\times(\mathbf{p_2}-\mathbf{p_3})}{\|(\mathbf{p_1}-\mathbf{p_3})\times(\mathbf{p_2}-\mathbf{p_3})\|}.$$

Similar calculations hold for the other normals. Consequently, $\mathbf{P}_t^{-1}\boldsymbol{\nu}_i$ are the normals at time $t = 0$, which stay constant in object-fixed coordinates.

By multiplying both sides of the force equation by $\mathbf{P}_t^{-1} = \mathbf{R}_t$ we get that

$$\mathbf{P}_t^{-1}\ddot{\mathbf{r}}_t = \tfrac{1}{m}\textstyle\sum_{i=0}^{3} f_i \mathbf{P}_t^{-1}\boldsymbol{\nu}_i.$$

With the appropriate substitutions, this equation becomes

$$(\mathbf{Y}_t^2 - \dot{\mathbf{Y}}_t)(\mathbf{r}_0 - \mathbf{u}_t) + 2\mathbf{Y}_t\dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t = \tfrac{1}{m}\textstyle\sum f_i\boldsymbol{\nu}_i,$$

which is the desired force equation in object-fixed coordinates. □

Proposition 5.1.2 *The Newtonian torque equation in object-fixed coordinates is:*

$$\mathbf{I}\dot{\boldsymbol{\Omega}} + m(\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) =$$
$$(\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \textstyle\sum_{i=0}^{3} f_i\mathbf{q}_i \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i\boldsymbol{\nu}_i.$$

Proof: Let $\hat{x}$ be a reference system fixed on the robot hand and $\hat{y}$ be a reference system fixed on the object. While $\hat{x}$ is inertial, $\hat{y}$ is not. Let $\mathbf{T}_t = \begin{pmatrix} \mathbf{P}_t & \mathbf{r}_t \\ 0 & 1 \end{pmatrix}$ be the matrix describing the motion of a particle relative to $\hat{x}$. Then $\mathbf{S}_t = \mathbf{T}_t^{-1} = \begin{pmatrix} \mathbf{R}_t & \mathbf{u}_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{P}_t^{-1} & -\mathbf{P}_t^{-1}\mathbf{r}_t \\ 0 & 1 \end{pmatrix}$ is the matrix describing the same motion relative to $\hat{y}$. Then if $\mathbf{Q}_t$ is the position of a point relative to $\hat{y}$ and $\mathbf{q}_t$ the position relative to $\hat{x}$, the two systems are related by $\mathbf{q}_t = \mathbf{P}_t\mathbf{Q}_t + \mathbf{r}_t$. Let $\boldsymbol{\Omega}$ be the angular velocity in $\hat{y}$ and $\boldsymbol{\omega}$ the angular velocity in $\hat{x}$. The two angular velocities are related by $\mathbf{P}\boldsymbol{\Omega} = \boldsymbol{\omega}$. By differentiating, we get the relationship between the velocities: $\dot{\mathbf{q}} = \mathbf{P}\dot{\mathbf{Q}} + \dot{\mathbf{P}}\mathbf{Q} + \dot{\mathbf{r}} = \mathbf{P}\dot{\mathbf{Q}} + (\dot{\mathbf{P}}\mathbf{P}^{-1})\mathbf{P}\mathbf{Q} + \dot{\mathbf{r}} = \mathbf{P}\dot{\mathbf{Q}} + \dot{\mathbf{P}}\mathbf{P}^{-1}(\mathbf{q} - \mathbf{r}) + \dot{\mathbf{r}} = \mathbf{P}\dot{\mathbf{Q}} + \boldsymbol{\omega} \times (\mathbf{q} - \mathbf{r}) + \dot{\mathbf{r}}$, since $\dot{\mathbf{P}}\mathbf{P}^{-1}\mathbf{u} = \boldsymbol{\omega} \times \mathbf{u}$ defines the angular velocity $\boldsymbol{\omega}$ in $\hat{x}$. Moreover, for any $\mathbf{U} \in \hat{y}$ we have $\mathbf{P}^{-1}\dot{\mathbf{P}}\mathbf{U} = \boldsymbol{\Omega} \times \mathbf{U}$, because $\dot{\mathbf{P}}\mathbf{P}^{-1}\mathbf{u} = \boldsymbol{\omega} \times \mathbf{u}$. Set $\mathbf{U} = \mathbf{P}^{-1}\mathbf{u}$, so $\mathbf{u} = \mathbf{P}\mathbf{U}$. Then $\dot{\mathbf{P}}\mathbf{U} = \boldsymbol{\omega} \times \mathbf{P}\mathbf{U}$, or $\mathbf{P}^{-1}\dot{\mathbf{P}}\mathbf{U} = \boldsymbol{\Omega} \times \mathbf{U}$.

Let $\mathbf{a}$ be the angular momentum in the inertial system $\hat{x}$ and $\mathbf{A}$ the angular momentum in the body-fixed system, $\hat{y}$. Let m denote the mass of the body.

Then for a one point body, $\mathbf{a} = m\mathbf{q} \times (\omega \times \mathbf{q}) = \mathbf{PA}$. By multiplying by $\mathbf{P}^{-1}$, $\mathbf{A} = m(\mathbf{P}^{-1}\mathbf{q} \times (\mathbf{P}^{-1}\omega \times \mathbf{P}^{-1}\mathbf{q})) = m((\mathbf{Q} + \mathbf{P}^{-1}\mathbf{r}) \times (\boldsymbol{\Omega} \times (\mathbf{Q} + \mathbf{P}^{-1}\mathbf{r}))) = m(\mathbf{Q} \times (\boldsymbol{\Omega} \times \mathbf{Q}) + \mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \mathbf{Q}) + \mathbf{Q} \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r}) + \mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r}))$.

For a continuous distribution of masses, we have

$\int_i m_i(\mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r})) = m\mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r})$, $\int_i m_i(\mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \mathbf{Q}_i)) = \mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times \int_i \mathbf{a}_i\mathbf{Q}_i) = \mathbf{P}^{-1}\mathbf{r} \times (\boldsymbol{\Omega} \times m\mathbf{Q}_{CM})$, $\int_i m_i(\mathbf{Q}_i \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r})) = m\mathbf{Q}_{CM} \times (\boldsymbol{\Omega} \times \mathbf{P}^{-1}\mathbf{r})$, and $\int_i m_i(\mathbf{Q}_i \times (\boldsymbol{\Omega} \times \mathbf{Q}_i)) = \mathbf{I}\boldsymbol{\Omega}$.

Let $\mathbf{n} = \dot{\mathbf{a}}$ be the total torque in $\hat{x}$. But $\mathbf{a} = \mathbf{PA}$ so $\dot{\mathbf{a}} = \dot{\mathbf{P}}\mathbf{A} + \mathbf{P}\dot{\mathbf{A}}$, or $\mathbf{P}^{-1}\dot{\mathbf{a}} = \mathbf{P}^{-1}\dot{\mathbf{P}}\mathbf{A} + \dot{\mathbf{A}}$. This implies that $\mathbf{P}^{-1}\mathbf{n} = \mathbf{P}^{-1}\dot{\mathbf{P}}\mathbf{A} + \dot{\mathbf{A}}$, in other words, $\mathbf{P}^{-1}\mathbf{n} = \boldsymbol{\Omega} \times \mathbf{A} + \dot{\mathbf{A}}$. Observe now that $\mathbf{n} = \mathbf{q} \times \mathbf{f}$ so $\mathbf{P}^{-1}\mathbf{n} = \mathbf{P}^{-1}(\mathbf{q} \times \mathbf{f}) = \mathbf{P}^{-1}\mathbf{q} \times \mathbf{P}^{-1}\mathbf{f} = (\mathbf{Q} + \mathbf{P}^{-1}\mathbf{r}) \times \mathbf{F} = \mathbf{Q} \times \mathbf{F} + \mathbf{P}^{-1}\mathbf{r} \times \mathbf{F}$. So the torque equation which is valid in $\hat{y}$ is:

$$\tfrac{d}{dt}\mathbf{A} = \mathbf{Q} \times \mathbf{F} + \mathbf{P}^{-1}\mathbf{r} \times \mathbf{F} + \mathbf{A} \times \boldsymbol{\Omega}, \text{ or}$$

$$\mathbf{A} = \mathbf{I}\boldsymbol{\Omega} + m(-\mathbf{u} \times (\boldsymbol{\Omega} \times \mathbf{Q}_{CM}) - \mathbf{Q}_{CM} \times (\boldsymbol{\Omega} \times \mathbf{u}) + \mathbf{u} \times (\boldsymbol{\Omega} \times \mathbf{u}))$$

Then the torque equations is

$\mathbf{I}\dot{\boldsymbol{\Omega}} + m(-\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{Q}_{CM}) - \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{Q}_{CM}) - \mathbf{Q}_{CM} \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) - \mathbf{Q}_{CM} \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t) + \dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) = (\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(-\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{Q}_{CM}) - \mathbf{Q}_{CM} \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \sum_{i=0}^{3} f_i\mathbf{q_i} \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i \times \boldsymbol{\nu}_i$.

If we choose the origin to be the center of mass, $\mathbf{Q}_{CM}$, several terms of the equation drop out. The new torque equation is:

$$\mathbf{I}\dot{\boldsymbol{\Omega}} + m(\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) =$$
$$(\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \sum_{i=0}^{3} f_i\mathbf{q_i} \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i\boldsymbol{\nu}_i.$$

□

Thus, the set of equations defining the constraint set for our problem is:

$$\det(\mathbf{Sq_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}) = 0 \tag{5.2}$$

$$\det(\mathbf{S}\mathbf{q_1}, \mathbf{p_2}, \mathbf{p_0}, \mathbf{p_3}) = 0 \tag{5.3}$$

$$\det(\mathbf{S}\mathbf{q_2}, \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_3}) = 0 \tag{5.4}$$

$$\det(\mathbf{q_3}, \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}) = 0 \tag{5.5}$$

$$(\boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{r} - \mathbf{u}_t)) - \dot{\boldsymbol{\Omega}}_t \times (\mathbf{r} - \mathbf{u}_t) + 2\boldsymbol{\Omega}_t \times \dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t = \frac{1}{m} \sum f_i \boldsymbol{\nu}_i. \tag{5.6}$$

$$\begin{aligned} &\mathbf{I}\dot{\boldsymbol{\Omega}} + m(\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) = \\ &(\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \sum_{i=0}^{3} f_i \mathbf{q_i} \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i \boldsymbol{\nu}_i. \end{aligned} \tag{5.7}$$

5.1.2 Defining the Configuration Space

The control of the tracking of a finger requires a plan for issuing acceleration commands. We can build the plan by knowing the instantaneous motion of the finger. Given some initial configuration for the grasped tetrahedron, and some desired axis of rotation, how should the tracking finger move to generate the desired instantaneous motion of the object? Our concern is only in the differential motion for now, and later in the paper we show how the control strategy can be used to generate any desired rotational motion. We assume that the fingers are not located near the boundary of their faces, and they are not in danger of sliding off. To find the tracking finger motion, what is needed is the configuration space, and a method of relating it to the tracking motion.

Theorem 5.1.3 *For a body-fixed reference frame, the configuration space is a manifold given by:*

$$M = \left\{ \left(\begin{array}{cc} \mathbf{R} & \mathbf{u}(\mathbf{R}) \\ 0 & 1 \end{array} \right) \middle| \ \mathbf{R} \in SO(3) \right\}, \quad \textit{where}$$

$$\mathbf{u}(\mathbf{R}) = -\frac{\langle \mathbf{R}\mathbf{q_0} - \mathbf{p_3}, \boldsymbol{\nu}_0 \rangle \mathbf{p}_0'}{\langle \mathbf{p}_0', \boldsymbol{\nu}_0 \rangle} - \frac{\langle \mathbf{R}\mathbf{q_1} - \mathbf{p_3}, \boldsymbol{\nu}_1 \rangle \mathbf{p}_1'}{\langle \mathbf{p}_1', \boldsymbol{\nu}_1 \rangle} - \frac{\langle \mathbf{R}\mathbf{q_2} - \mathbf{p_3}, \boldsymbol{\nu}_2 \rangle \mathbf{p}_2'}{\langle \mathbf{p}_2', \boldsymbol{\nu}_2 \rangle}.$$

Proof: By taking into consideration the rotational and translational components of $\mathbf{S}$, each constraint can be rewritten as

$$\det(\mathbf{R}\mathbf{q_i} - \mathbf{p_3} + \mathbf{u}, \mathbf{p'_j}, \mathbf{p'_k}) = 0,$$

where (i, j, k) is a cyclic permutation of $(0, 1, 2)$. This equation can be expressed in terms of vector operations as:

$$\langle \mathbf{R}\mathbf{q_i} - \mathbf{p_3} + \mathbf{u}, \mathbf{p'_j} \times \mathbf{p'_k} \rangle = 0.$$

The vectors $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}$ define a non-degenerate face and thus are linearly independent. We can express $\mathbf{u} = \alpha\mathbf{p_0} + \beta\mathbf{p_1} + \gamma\mathbf{p_2}$. We get that $\alpha = -\frac{\langle \mathbf{R}\mathbf{q_0} - \mathbf{p_3}, \boldsymbol{\nu}_0 \rangle}{\langle \mathbf{p'_0}, \boldsymbol{\nu}_0 \rangle}$, $\beta = -\frac{\langle \mathbf{R}\mathbf{q_1} - \mathbf{p_3}, \boldsymbol{\nu}_1 \rangle}{\langle \mathbf{p'_1}, \boldsymbol{\nu}_1 \rangle}$, and $\gamma = -\frac{\langle \mathbf{R}\mathbf{q_2} - \mathbf{p_3}, \boldsymbol{\nu}_2 \rangle}{\langle \mathbf{p'_2}, \boldsymbol{\nu}_2 \rangle}$. □

Theorem 5.1.4 *For a space-fixed reference frame, the configuration space is a manifold given by:*

$$M' = \left\{ \begin{pmatrix} \mathbf{P} & \mathbf{v}(\mathbf{P}) \\ 0 & 1 \end{pmatrix} \middle| \; \mathbf{P} = \mathbf{R}^{-1} \in \mathbf{SO}(3) \right\}, \quad \textit{where}$$

$$\mathbf{v}(\mathbf{P}) = \frac{\langle \mathbf{q_0} - \mathbf{P}\mathbf{p_3}, \mathbf{P}\mathbf{p'_1} \times \mathbf{P}\mathbf{p'_2} \rangle}{\langle \mathbf{P}\mathbf{p'_0}, \mathbf{P}\boldsymbol{\nu}_0 \rangle}\mathbf{P}\mathbf{p'_0} + \frac{\langle \mathbf{q_1} - \mathbf{P}\mathbf{p_3}, \mathbf{P}\mathbf{p'_2} \times \mathbf{P}\mathbf{p'_0} \rangle}{\langle \mathbf{P}\mathbf{p'_1}, \mathbf{P}\boldsymbol{\nu}_1 \rangle}\mathbf{P}\mathbf{p'_1} + \frac{\langle \mathbf{q_2} - \mathbf{P}\mathbf{p_3}, \mathbf{P}\mathbf{p'_0} \times \mathbf{P}\mathbf{p'_1} \rangle}{\langle \mathbf{P}\mathbf{p'_2}, \mathbf{P}\boldsymbol{\nu}_2 \rangle}\mathbf{P}\mathbf{p'_2}.$$

Proof: Let $\mathbf{P} = \mathbf{R}^{-1}$ and let $\mathbf{v} = -\mathbf{R}^{-1}\mathbf{u}$. By substituting the expression for $\mathbf{u}$ in the formula for $\mathbf{v}$ the result follows. □

To understand the connection between M and M', consider $\mathbf{T}_t$, a motion of the tetrahedron from time $t = 0$ to time t that preserves the motion constraints. The inverse of this motion, denoted by $\mathbf{S}_t = \mathbf{T}_t^{-1}$ is also a motion between valid configurations. The formula for $\mathbf{u}_t$ gives the translation compatible with motion within M, as a function of rotation. By noting that $\mathbf{T}_t \in M' \Leftrightarrow \mathbf{T}_t^{-1} \in M \Leftrightarrow \mathbf{T}_t \in M^{-1}$, we can see that $M' = M^{-1}$.

5.1.3 Defining the Control

Having obtained an equational expression for the configuration space makes the control aspect of the finger tracking problem easier to tackle. The desired trajectory for the motion of the object is a curve in this configuration space. To actually generate this trajectory by finger tracking, the configuration space must be related to the force space of the hand and the acceleration space of the object. The acceleration constraints associated with motion compatible with staying in M are given next.

Proposition 5.1.5 *The acceleration constraints associated with motion compatible with staying in M have the form:*

$$det(\ddot{\mathbf{q}}_{\mathbf{i}t}, \mathbf{p_j} - \mathbf{p_l}, \mathbf{p_k} - \mathbf{p_l}) =$$
$$det(\dot{\boldsymbol{\Omega}}_t \times (\mathbf{q}_{\mathbf{i}t} - u_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q}_{\mathbf{i}t} - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t, \mathbf{p_j} - \mathbf{p_l}, \mathbf{p_k} - \mathbf{p_l}) = 0.$$

Proof: Recall that there are three constraint equations defining the configuration space, of the form $\det(\mathbf{q}_{\mathbf{i}t}, \mathbf{p_j}, \mathbf{p_k}) = 0$, where $\mathbf{q}_{\mathbf{i}t}$ is the position of $\mathbf{q_i}$ at time t, which is given by $\mathbf{R}_t\mathbf{q_i} + \mathbf{u}_t$. The velocity and acceleration constraints that correspond to this equation are $\det(\dot{\mathbf{q}}_t, \mathbf{p_j}, \mathbf{p_k}) = 0$, and $\det(\ddot{\mathbf{q}}_t, \mathbf{p_j}, \mathbf{p_k}) = 0$. Since $\mathbf{q}_t = \mathbf{R}_t\mathbf{q_0} + \mathbf{u}_t$, it follows that $\mathbf{q_0} = \mathbf{R}_t^{-1}(\mathbf{q}_t - \mathbf{u}_t)$. Then $\dot{\mathbf{q}}_t = \dot{\mathbf{R}}_t\mathbf{q_0} + \dot{\mathbf{u}}_t$, and this implies that $\dot{\mathbf{q}}_t = \dot{\mathbf{R}}_t\mathbf{R}_t^{-1}(\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t = \mathbf{Y}_t(\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t = \mathbf{Y}_t\mathbf{q}_t + \dot{\mathbf{u}}_t - \mathbf{Y}_t\mathbf{u}_t$. Furthermore, $\ddot{\mathbf{q}}_t = \dot{\mathbf{Y}}_t\mathbf{q}_t + \mathbf{Y}_t\dot{\mathbf{q}}_t + \ddot{\mathbf{u}}_t - \dot{\mathbf{Y}}_t\mathbf{u}_t - \mathbf{Y}_t\dot{\mathbf{u}}_t = \dot{\mathbf{Y}}_t + \mathbf{Y}_t^2(\mathbf{q}_t - \mathbf{u}_t) + \ddot{\mathbf{u}}_t$. □

The acceleration constraints, combined with the force and torque equation for this instantiation of manipulation yield a system of equations that governs the motion of the problem. Given a tetrahedron and a robot hand with four fingers, three of which are fixed and one free to track, $f : \mathcal{B} \times \mathcal{C} \times \mathcal{P}(\mathcal{M}) \rightarrow \mathcal{F}$ is given by the following system of equations:

$$(\boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{r} - \mathbf{u}_t)) - \dot{\boldsymbol{\Omega}}_t \times (\mathbf{r} - \mathbf{u}_t) + 2\boldsymbol{\Omega}_t \times \dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t = \frac{1}{m}\sum f_i\boldsymbol{\nu}_i. \quad (5.8)$$

$$\mathbf{I}\dot{\boldsymbol{\Omega}} + m(\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) =$$

$$(\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \sum_{i=0}^{3} f_i \mathbf{q_i} \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i \boldsymbol{\nu}_i. \qquad (5.9)$$

$$\det(\dot{\boldsymbol{\Omega}}_t \times (\mathbf{q_0}_t - u_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q_0}_t - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t, \mathbf{p}_1', \mathbf{p}_2') = 0. \qquad (5.10)$$

$$\det(\dot{\boldsymbol{\Omega}}_t \times (\mathbf{q_1}_t - u_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q_1}_t - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t, \mathbf{p}_2', \mathbf{p}_0') = 0. \qquad (5.11)$$

$$\det(\dot{\boldsymbol{\Omega}}_t \times (\mathbf{q_2}_t - u_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q_2}_t - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t, \mathbf{p}_0', \mathbf{p}_1') = 0. \qquad (5.12)$$

$$\det(\overline{\mathbf{q_3}}, \overline{\mathbf{p_0}}, \overline{\mathbf{p_1}}, \overline{\mathbf{p_2}}) = 0 \qquad (5.13)$$

Equations 5.8 and 5.9 are vectorial. Equations 5.10, 5.11, and 5.12 are scalar and give the acceleration constraints. Equation 5.13 is another scalar equation that binds the location of the tracking finger $\mathbf{q_3}$ to the plane defined by vertices $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}$. Thus, we have a system of 10 equations. This system can be used to answer various questions. In particular, it can be used to find how to track finger $\mathbf{q_3}$ in order to generate some desired object reorientation. When the axis of rotation is given as a function of time, this function also defines the angular acceleration of the body. The angular acceleration, together with the geometry of the object and the sensed velocities are the known parameters of the system. Their values can be used to solve for the translational acceleration, the force magnitudes, the tracking direction for $\mathbf{q}_3$, and further for the new contact locations. The system can also be used to predict the motion behavior of a grasped object, when the tracking finger moves along some specified trajectory, and the fingers exert given forces.

Theorem 5.1.6 *Given a tetrahedron, a robot hand with at least four fingers operating with three constraining fingers and one tracking finger, and a curve γ in the configuration space M, at any point in time the tracking motion required to generate γ is given by a generally solvable, square linear system.*

Proof: Choose the origin of the reference frame to be the center of mass At any point in time the angular acceleration can be obtained by differentiating the desired trajectory γ. If $\gamma(t) = \begin{pmatrix} \mathbf{R}_t & \mathbf{u}_t \\ 0 & 1 \end{pmatrix}$ then $\dot{\mathbf{\Omega}} = \ddot{\mathbf{R}}_t\mathbf{R}_t^{-1} - (\dot{\mathbf{R}}_t\mathbf{R}_t^{-1})^2$. To show that the control of finger tracking is a linear problem, we start with the system of 10 equations given above and we transform it to emphasize the unknown quantities. For this application, they are the force magnitudes, the direction of motion for the tracking finger and the translational acceleration. The idea of the transformation is to combine the values for $\dot{\mathbf{\Omega}}$ and $\ddot{\mathbf{u}}$ given by the first two equations with the constraint equations, and then to solve for the unknowns.

First of all, note that $\mathbf{u} \times (\dot{\mathbf{\Omega}} \times \mathbf{u}) = |\ \mathbf{u}\ |^2\dot{\mathbf{\Omega}} - \langle \mathbf{u}, \dot{\mathbf{\Omega}} \rangle \mathbf{u} = (|\ \mathbf{u}\ |^2\mathcal{I} - \mathbf{u}\mathbf{u}^T)\dot{\mathbf{\Omega}}$, where $\mathcal{I}$ is the identity matrix. Let $\mathbf{I}_t = \mathbf{I} + m(|\ \mathbf{u}\ |^2)\mathcal{I} - \mathbf{u}\mathbf{u}^T)$. This is the coefficient for $\dot{\mathbf{\Omega}}$ in the torque equation. Let $\mathbf{c_1} = (\mathbf{I\Omega}) \times \mathbf{\Omega} + m(\mathbf{u} \times (\mathbf{\Omega} \times \mathbf{u})) \times \mathbf{\Omega} - m\dot{\mathbf{u}} \times (\mathbf{\Omega} \times \mathbf{u}) - m\mathbf{u} \times (\mathbf{\Omega} \times \dot{\mathbf{u}})$ and let $\mathbf{c_2} = -\mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{u}) + 2\mathbf{\Omega} \times \dot{\mathbf{u}}$. We group the terms of the two equations containing velocities, because from an application viewpoint, these are values that can be determined after integration. With this notation, and by substituting the expressions for $\dot{\mathbf{\Omega}}$ and $\ddot{\mathbf{u}}$ in the constraint equations, the equations of the system can be rewritten as:

1. $\dot{\mathbf{\Omega}} = \mathbf{I}_t^{-1}\mathbf{c_1} + \sum_{i=0}^{3} f_i \mathbf{I}_t^{-1}((\mathbf{q_i} - \mathbf{u}) \times \boldsymbol{\nu}_i)$

2. $\ddot{\mathbf{u}} = \mathbf{c_2} + \mathbf{I}_t^{-1}\mathbf{c_1} \times \mathbf{u} + \sum_{i=0}^{3} f_i(\mathbf{I}_t^{-1}((\mathbf{q_i} - \mathbf{u}) \times \boldsymbol{\nu}_i) \times \mathbf{u} - \frac{\boldsymbol{\nu}_i}{m})$

3. $\langle \mathbf{c_2} + \mathbf{I}_t^{-1}\mathbf{c_1} \times \mathbf{q_0} + \sum_{i=0}^{3} f_i(\mathbf{I}_t^{-1}((\mathbf{q_i} - \mathbf{u}) \times \boldsymbol{\nu}_i) \times \mathbf{q_0} - \frac{\boldsymbol{\nu}_i}{m}) +$
   $\mathbf{\Omega} \times (\mathbf{\Omega} \times (\mathbf{q_0} - \mathbf{u})), \mathbf{p}'_1 \times \mathbf{p}'_2 \rangle = 0$

4. $\langle \mathbf{c_2} + \mathbf{I}_t^{-1}\mathbf{c_1} \times \mathbf{q_1} + \sum_{i=0}^{3} f_i(\mathbf{I}_t^{-1}((\mathbf{q_i} - \mathbf{u}) \times \boldsymbol{\nu}_i) \times \mathbf{q_1} - \frac{\boldsymbol{\nu}_i}{m}) +$
   $\mathbf{\Omega} \times (\mathbf{\Omega} \times (\mathbf{q_1} - \mathbf{u})), \mathbf{p}'_2 \times \mathbf{p}'_0 \rangle = 0$

5. $\langle \mathbf{c_2} + \mathbf{I}_t^{-1}\mathbf{c_1} \times \mathbf{q_2} + \sum_{i=0}^{3} f_i(\mathbf{I}_t^{-1}((\mathbf{q_i} - \mathbf{u}) \times \boldsymbol{\nu}_i) \times \mathbf{q_2} - \frac{\boldsymbol{\nu}_i}{m}) +$
   $\mathbf{\Omega} \times (\mathbf{\Omega} \times (\mathbf{q_2} - \mathbf{u})), \mathbf{p}'_0 \times \mathbf{p}'_1 \rangle = 0$

6. $\det(\overline{\mathbf{q_3}}, \overline{\mathbf{p_0}}, \overline{\mathbf{p_1}}, \overline{\mathbf{p_2}}) = 0.$

Let us concentrate for now on the constraint equations. We can group together the terms that do not contain the unknowns. Let $\mathbf{b_i} = -\langle \mathbf{c_2} + \mathbf{I}_t^{-1}\mathbf{c_1} \times \mathbf{q_i} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times (\mathbf{q_i} - \mathbf{u})), \mathbf{p'_j} \times \mathbf{p'_k}\rangle$, for each even permutation (i, j, k) of $(0, 1, 2)$. Then each constraint equation is given by:

$$\mathbf{b_i} = \sum_{j=0}^{3} f_j \langle (\mathbf{I}_t^{-1}((\mathbf{q_j} - \mathbf{u}) \times \boldsymbol{\nu}_j) \times \mathbf{q_i} - \frac{\boldsymbol{\nu}_j}{m}), \mathbf{p'_k} \times \mathbf{p'_l}\rangle.$$

The terms of the right hand side contain two different types of unknowns. The terms indexed by $j = 0, 1, 2$ have only the force magnitudes as unknowns. The term indexed by $j = 3$ has the force magnitude, as well as the location of the contact point as unknowns. To process further the first type of terms, let $\mathbf{E_j} = \mathbf{I}_t^{-1}((\mathbf{q_j} - \mathbf{u}) \times \boldsymbol{\nu}_j)$, for $j = 0, 1, 2$. The matrix whose columns are $\mathbf{E_0}, \mathbf{E_1}, \mathbf{E_2}$ is called $\mathbf{E}$. Then $\sum_{j=0}^{2} f_j \mathbf{I}_t^{-1}((\mathbf{q_j} - \mathbf{u}) \times \boldsymbol{\nu}_j) = \mathbf{E}\begin{pmatrix} f_0 \\ f_1 \\ f_2 \end{pmatrix}$. In processing the other term of the sum, we want to emphasize that the unknowns are the coordinates of $\mathbf{q_3}$ which is constrained to be on the plane of $\mathbf{p_0}, \mathbf{p_1}$, and $\mathbf{p_2}$ (Equation 6,) as well as f_3. Change the coordinates of $\mathbf{q_3}$ to barycentric coordinates. If s_0, s_1, and s_2 are the barycentric coordinates for $\mathbf{q_3}$, (with $s_0 + s_1 + s_2 = 1$.) then $\mathbf{q_3} = s_0(\mathbf{p_0} - \mathbf{p_2}) + s_1(\mathbf{p_1} - \mathbf{p_2}) + \mathbf{p_2}$. By substituting this expression for $\mathbf{q_3}$ in $\mathbf{I}_t^{-1} f_3((\mathbf{q_3} - \mathbf{u}) \times \boldsymbol{\nu}_3)$, and by denoting $\mathbf{D_0} = \mathbf{I}_t^{-1}((\mathbf{p_0} - \mathbf{p_2}) \times \boldsymbol{\nu}_3), \mathbf{D_1} = \mathbf{I}_t^{-1}((\mathbf{p_1} - \mathbf{p_2}) \times \boldsymbol{\nu}_3), \mathbf{D_2} = \mathbf{I}_t^{-1}((\mathbf{p_2} - \mathbf{u}) \times \boldsymbol{\nu}_3)$, and $\mathbf{D}$ the matrix whose columns are $\mathbf{D_0}, \mathbf{D_1}, \mathbf{D_2}$, we get that $\mathbf{I}_t^{-1} f_3((\mathbf{q_3} - \mathbf{u}) \times \boldsymbol{\nu}_3) = \mathbf{D}\begin{pmatrix} f_3 s_0 \\ f_3 s_1 \\ f_3 \end{pmatrix}$. Let $\begin{pmatrix} f_3 s_0 \\ f_3 s_1 \\ f_3 \end{pmatrix} = \mathbf{y}$ and let $\begin{pmatrix} f_0 \\ f_1 \\ f_2 \end{pmatrix} = \mathbf{x}$. With this notation, for each even permutation (i, j, k) of $(0, 1, 2)$ the constraint equations are:

$$\mathbf{b_i} = (\mathbf{q_i} \times (\mathbf{p'_k} \times \mathbf{p'_l}))^T (\mathbf{Ex} + \mathbf{Dy}) - \tfrac{y_2}{m}\langle \boldsymbol{\nu}_i, \mathbf{p'_k} \times \mathbf{p'_l}\rangle -$$
$$\tfrac{1}{m}\begin{pmatrix} \langle \boldsymbol{\nu}_i, \mathbf{p'_k} \times \mathbf{p'_l}\rangle & \langle \boldsymbol{\nu}_i, \mathbf{p'_k} \times \mathbf{p'_l}\rangle & \langle \boldsymbol{\nu}_i, \mathbf{p'_k} \times \mathbf{p'_l}\rangle \end{pmatrix}\mathbf{x}.$$

We can express the system consisting of three such equations as a matrix system.

Let $\mathbf{H} = \begin{pmatrix} (\mathbf{q}_0 \times (\mathbf{p}'_1 \times \mathbf{p}'_2))^T \\ (\mathbf{q}_1 \times (\mathbf{p}'_2 \times \mathbf{p}'_0))^T \\ (\mathbf{q}_2 \times (\mathbf{p}'_0 \times \mathbf{p}'_1))^T \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$, $\mathbf{K} = \begin{pmatrix} (\mathbf{p}'_1 \times \mathbf{p}'_2)^T \\ (\mathbf{p}'_2 \times \mathbf{p}'_0)^T \\ (\mathbf{p}'_0 \times \mathbf{p}'_1)^T \end{pmatrix}$, $\mathbf{N} = \mathbf{K}\begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}$, $\mathbf{G} = \mathbf{K}\boldsymbol{\nu}_3$. Then the system of motion is:

$$\begin{cases} \mathbf{b} = (\mathbf{HE} - \frac{\mathbf{N}}{\mathbf{m}})\mathbf{x} + \mathbf{HDy} - \frac{y_2}{m}\mathbf{G} \\ \dot{\boldsymbol{\Omega}} = \mathbf{I}_t^{-1}\mathbf{c}_1 + \mathbf{Ex} + \mathbf{Dy} \end{cases}$$

This shows clearly that we can determine the unknowns by solving a linear system. The explicit solution can be obtained after some algebraic manipulation. What follows is that

$$\dot{\boldsymbol{\Omega}} - \mathbf{I}_t^{-1}\mathbf{c}_1 - \mathbf{EN}^{-1}m(-\mathbf{b} + \mathbf{H}\dot{\boldsymbol{\Omega}} - \mathbf{HI}_t^{-1}\mathbf{c}_1), \text{ or}$$

$$\mathbf{Dy} - \mathbf{EN}^{-1}\mathbf{G}y_2 = (\mathbf{D} - \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{EN}^{-1}\mathbf{G} \end{pmatrix})\mathbf{y} = \mathbf{Ay}. \tag{5.14}$$

and

$$\mathbf{x} = \mathbf{N}^{-1}(m\mathbf{H}\dot{\boldsymbol{\Omega}} - m\mathbf{b} - m\mathbf{HI}_t^{-1}\mathbf{c}_1 - y_2\mathbf{G}). \tag{5.15}$$

As long as the matrix $\mathbf{A}$ is not singular, there is a unique solution for $\mathbf{q_3}$ and f_3. When $\mathbf{N}$ is not singular, namely when the normals are linearly independent, there is also a unique solution for the force magnitudes f_0, f_1, f_2. □

Theorem 5.1.6 gives immediately an algorithm for the instantaneous control of the tracking finger. The values obtained by solving the linear system are the parameters needed by the hybrid-controller of the tracking finger to actually generate the desired behavior.

We have to point out that although the system of motion may give a unique solution, the solution may not be feasible. This happens when at least one of the force magnitudes is negative. The infeasible solutions can be eliminated by appending the system with the appropriate inequalities that enforce $f_0 > 0, f_1 > 0, f_2 > 0, f_3 > 0$. The properties of the appended system may be analyzed with standard linear programming techniques.

Notation: P is the polyhedron and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$ the fingers of the hand.

$\hat{x}$ is a coordinate system fixed on P. $\hat{x}.z$ is aligned with $\boldsymbol{\nu}_3$.

Input: $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}, \mathbf{I}, m$

current $\mathbf{q_3}, \boldsymbol{\Omega}, \ddot{\mathbf{u}}, \dot{\mathbf{u}}, \mathbf{u}, f_0, f_1, f_2, f_3$

desired $\dot{\boldsymbol{\Omega}}$

Build the system in Theorem 5.1.6.

Solve for new f_0, f_1, f_2, f_3 and $\mathbf{q_3}$.

Position control $\mathbf{f_3}$ in $\hat{x}.x$ and $\hat{x}.y$ to the new location $\mathbf{q_3}$.

Force control $\mathbf{f_3}$ in $\hat{x}.z$ to exert new f_3.

Force control $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}$ to exert new f_0, f_1, f_2.

Figure 5.1: Algorithm 5.1 for Instantaneous Finger Tracking

The result of the theorem contains the main motivation for why this is a good scheme to generate rotations by finger tracking and gives an explicit solution. Rotations by finger tracking are feasible and easy computationally, since they involve solving a small linear system at each time step. If the system is non-singular, then it gives the differential direction for the tracking finger. When the matrix of the system is not invertible, the system is singular and it corresponds to a configuration for which the desired axis of rotation is not available. Not all the axes of rotation are available at every time step. The following result quantifies this observation.

Corollary 5.1.7 *At each point in time, the available axes of rotation form a convex set. This convex set is obtained by taking the convex hull of the axis that result by moving the tracking finger to the vertices of its face.*

An axis is available if the system of Theorem 5.1.6 has a solution. When the force magnitudes resulting from the system are positive, then the available axis is also feasible. If an axis is available for the current configuration of the hand-object system, then the system of Theorem 5.1.6 can be employed to determine the necessary finger forces and the tracking direction. This is a strategy that combines position control with force control. By incorporating force control in the algorithm, we allow for compliant motion constrained by the task geometry. Controllers based on mixing force and position control are called hybrid, and have been studied and analyzed in [Mas86,RC81].

Thus far, we have showed how to derive a system of equations that determines the forces and tracking curves needed to generate a desired manipulation. The system was shown to be linear, but for realistic applications, we need to have a finer understanding of the structure of the solution set. In particular, we are interested in the sign of the unknowns. An acceptable solution must have positive force values and tracking barycentric coordinates that do not exceed the face boundaries.

To find the generic singularities [BR], consider the matrices that get inverted in the derivation process. If all the quantities are in general position, we can assume that $\mathbf{N}^{-1}$ and $\mathbf{I}_t^{-1}$ are non-singular. Let $\mathbf{D} = \mathbf{I}_t^{-1}\mathbf{D}$ and $\mathbf{E} = \mathbf{I}_t^{-1}\mathbf{E}$. The matrix of the system that gets inverted is $\left(\mathbf{D} - \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{E}\mathbf{N}^{-1}\mathbf{G} \end{pmatrix}\right)$. Since each column of $\mathbf{D}$ is perpendicular to the normal $\boldsymbol{\nu}_3$, the generic singularities of this matrix occur when $\mathbf{E}\mathbf{N}^{-1}\mathbf{G}$ is perpendicular to $\boldsymbol{\nu}_3$. Since $\mathbf{G} = \mathbf{K}\boldsymbol{\nu}_3$, we have that $\mathbf{E}\mathbf{N}^{-1}\mathbf{G} = \mathbf{E}\mathbf{N}^{-1}\mathbf{K}\boldsymbol{\nu}_3$. Thus, the condition for $\boldsymbol{\nu}_3$ to be perpendicular to $\mathbf{E}\mathbf{N}^{-1}\mathbf{G}$ can be written in terms of the inner product as:

$$\left\langle \boldsymbol{\nu}_3, \begin{pmatrix} \mathbf{R}_t\mathbf{q}_0 \times \boldsymbol{\nu}_0 & \mathbf{R}_t\mathbf{q}_1 \times \boldsymbol{\nu}_1 & \mathbf{R}_t\mathbf{q}_2 \times \boldsymbol{\nu}_2 \end{pmatrix} \right\rangle = 0.$$

Note that each of the $\mathbf{R}_t\mathbf{q}_\mathbf{i}$ terms is in fact $\mathbf{R}_t\mathbf{q}_\mathbf{i} + \mathbf{u} - \mathbf{R}_t\mathbf{q}_\mathbf{m} - \mathbf{u}$, where $\mathbf{q}_\mathbf{m}$ denotes the center of mass. In the system with the body-fixed coordinates, $\mathbf{R}_t\mathbf{q}_\mathbf{m}$ stay the same, and is the origin, throughout the manipulation.

We now wish to identify the configurations where the force calculation is singular.

Proposition 5.1.8 *The condition for $\boldsymbol{\nu}_3$ to be perpendicular to* $\mathbf{EN}^{-1}\mathbf{G}$ *is given by* $\langle \mathbf{V}, \mathbf{QV} \rangle = 0$, *where*

$$\mathbf{V} = \frac{1}{det(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2)} \begin{pmatrix} \boldsymbol{\nu}_1 \times \boldsymbol{\nu}_2 & \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_0 & \boldsymbol{\nu}_0 \times \boldsymbol{\nu}_1 \end{pmatrix} \boldsymbol{\nu}_3$$

and

$$\mathbf{Q} = \begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^t \begin{pmatrix} \mathbf{R_t q_0} \times \boldsymbol{\nu}_0 & \mathbf{R_t q_1} \times \boldsymbol{\nu}_1 & \mathbf{R_t q_2} \times \boldsymbol{\nu}_2 \end{pmatrix}.$$

Proof: The condition can be written as

$$\boldsymbol{\nu}_3{}^t \begin{pmatrix} \mathbf{R_t q_0} \times \boldsymbol{\nu}_0 & \mathbf{R_t q_1} \times \boldsymbol{\nu}_1 & \mathbf{R_t q_2} \times \boldsymbol{\nu}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^{-1} \boldsymbol{\nu}_3.$$

By inserting the identity matrix $\left(\begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^{-1}\right)^t \begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^t$, and observing that $\begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^{-1} = \frac{1}{\det(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2)} \begin{pmatrix} \boldsymbol{\nu}_1 \times \boldsymbol{\nu}_2 & \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_0 & \boldsymbol{\nu}_0 \times \boldsymbol{\nu}_1 \end{pmatrix}$ the result follows. □

Proposition 5.1.8 essentially says that the perpendicularity condition is given by the quadric form associated with a 3×3 matrix. The function that gives this condition is $f : R^9 \to R$, with $f(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}) = \langle \mathbf{V}, \mathbf{QV} \rangle$. The zeros of f are the singular configurations for the tracking problem.

Let the zero set of f be called $\mathbf{Q_z}$. Since the normals $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2$ are independent, they form a basis. Let $\{\mathbf{w_0}, \mathbf{w_1}, \mathbf{w_2}\}$ be a dual basis. In other words, $\langle \mathbf{w_i}, \boldsymbol{\nu}_j \rangle$ is 0 if $i \neq j$ and 1 if $i = j$.

Theorem 5.1.9 *The zero set of* $f : R^9 \to R$, *with* $f(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}) = \langle \mathbf{V}, \mathbf{QV} \rangle$ *is given by:*

$$\mathbf{Q_z} = \left\{ (\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}) \mid \frac{\langle \mathbf{q_1} - \mathbf{q_0}, \mathbf{w_2} \rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_2} \rangle} + \frac{\langle \mathbf{q_2} - \mathbf{q_1}, \mathbf{w_0} \rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_0} \rangle} + \frac{\langle \mathbf{q_0} - \mathbf{q_2}, \mathbf{w_1} \rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_1} \rangle} = 0 \right\}.$$

Proof: From the definition of $\mathbf{Q}$ it follows that $\begin{pmatrix} x & y & z \end{pmatrix} \mathbf{Q} \begin{pmatrix} x & y & z \end{pmatrix}^t = xy\langle \mathbf{q_1} - \mathbf{q_0}, \boldsymbol{\nu}_0 \times \boldsymbol{\nu}_1 \rangle + zx\langle \mathbf{q_0} - \mathbf{q_2}, \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_0 \rangle + yz\langle \mathbf{q_2} - \mathbf{q_1}, \boldsymbol{\nu}_1 \times \boldsymbol{\nu}_2 \rangle$. Since $\{\mathbf{w_0}, \mathbf{w_1}, \mathbf{w_2}\}$ is a dual basis to $\{\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2\}$ it follows that $\boldsymbol{\nu}_i \times \boldsymbol{\nu}_j = \det(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2)\mathbf{w_k}$ for (i, j, k) a permutation of $(0, 1, 2)$. Moreover, it follows that $V = \begin{pmatrix} \langle \boldsymbol{\nu}_3 \mathbf{w_0} \rangle & \langle \boldsymbol{\nu}_3 \mathbf{w_1} \rangle & \langle \boldsymbol{\nu}_3 \mathbf{w_2} \rangle \end{pmatrix}^t$. By using this value for V and by denoting $\det(\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2) = d$ we have that the expression for $f(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})$ is given by $-d\langle \boldsymbol{\nu}_3, \mathbf{w_0}\rangle\langle \boldsymbol{\nu}_3, \mathbf{w_1}\rangle\langle \boldsymbol{\nu}_3, \boldsymbol{w}_2\rangle \left(\frac{\langle \mathbf{q_1} - \mathbf{q_0}, \mathbf{w_2}\rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_2}\rangle} + \frac{\langle \mathbf{q_2} - \mathbf{q_1}, \mathbf{w_0}\rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_0}\rangle} + \frac{\langle \mathbf{q_0} - \mathbf{q_2}, \mathbf{w_1}\rangle}{\langle \boldsymbol{\nu}_3, \mathbf{w_1}\rangle} \right)$. The formula for the zero set thus follows. □

We observe that f is a 9-dimensional linear function. Thus, its zero set is 8-dimensional, or, alternatively, Q_z has codimension 1. The significance of the closed-form description of the zero set of f is that for any configuration $(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})$, this allows for the computation of the minimum rotation into a singular configuration. In manipulation strategies, singular configurations of the fixed fingers (or equivalently, the constraints) $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$ must be avoided, since they correspond to configurations where the fingers are required to exert infinitely large forces. A large minimum rotation $\mathbf{R}$ that takes $(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})$ into a singular configuration is the expression of good manipulability from the initial grasp.

The following theorem gives a lower bound on the rotation that moves the fixed contacts $(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})$ into the null set [BR].

Theorem 5.1.10 *The smallest angle that could rotate (q_0, q_1, q_2) into Q_z is given by:*

$$cos\theta = \frac{\bar{f}(\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})}{\|\bar{\mathbf{N}}\|\|\mathbf{q}\|},$$

where $N \in R^9$ is $\left(\frac{\mathbf{w_2}}{\langle \nu_3, \mathbf{w_2}\rangle}, \frac{\mathbf{w_0}}{\langle \nu_3, \mathbf{w_0}\rangle}, \frac{\mathbf{w_1}}{\langle \nu_3, \mathbf{w_1}\rangle}\right)$ *and* $\mathbf{q} = (\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2})$.

5.1.4 Instantaneous Rotations of Arbitrary Polyhedra

In generating the tracking direction and the forces for infinitesimal motions of the tetrahedra, the geometry of the object is relevant only in determining the equations of the configuration space and of the acceleration constraints for motion compatible with staying within the configuration space. The special geometry of the tetrahedron was not used in any fundamental way. The important components were the equations of the faces of the tetrahedron that were assigned to contact the grasping fingers. These equations were defined in terms of the vertices of the tetrahedron. This observation suggests that the same process can be used to define the configuration space for an arbitrary polyhedron, grasped by a hand with three fixed fingers and a tracking finger. Four faces of the polyhedron that are in general position are selected and assigned to contact the grasping fingers. For each face, three of its vertices are selected to define the plane of the face. The constraint equations are defined in terms of the chosen vertices.

Let P be a polyhedron and let $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_1}$ be contact points assigned to faces with linearly independent normals $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2$. Let $\mathbf{p_0}$ be a vertex on the face whose normal is $\boldsymbol{\nu}_0$, $\mathbf{p_1}$ a vertex on the face whose normal is $\boldsymbol{\nu}_1$, and $\mathbf{p_2}$ a vertex on the face whose normal is $\boldsymbol{\nu}_2$.

Theorem 5.1.11 *For a polyhedron kinematically constrained by three fixed points on faces with linearly independent normals, the configuration space is a manifold given by:*

$$M = \left\{ \begin{pmatrix} \mathbf{R} & \mathrm{u}(\mathbf{R}) \\ 0 & 1 \end{pmatrix} \middle| \; \mathbf{R} \in \mathbf{SO}(3) \right\}, \quad \textit{where}$$

$$\mathrm{u}(\mathbf{R}) = \alpha\boldsymbol{\nu}_0 + \beta\boldsymbol{\nu}_1 + \gamma\boldsymbol{\nu}_2, \quad \textit{and}$$

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = -\left(\begin{pmatrix} \boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 \end{pmatrix}^2\right)^{-1} \begin{pmatrix} \langle \mathbf{R}\mathbf{q_0} - \mathbf{p_0}, \boldsymbol{\nu}_0 \rangle \\ \langle \mathbf{R}\mathbf{q_1} - \mathbf{p_1}, \boldsymbol{\nu}_1 \rangle \\ \langle \mathbf{R}\mathbf{q_2} - \mathbf{p_2}, \boldsymbol{\nu}_2 \rangle \end{pmatrix}.$$

Proof: The three constraint equations can be written as $\langle \mathbf{S}\mathbf{q_0} - \mathbf{p_0}, \boldsymbol{\nu}_0 \rangle = 0, \langle \mathbf{S}\mathbf{q_1} - \mathbf{p_1}, \boldsymbol{\nu}_1 \rangle = 0$, and $\langle \mathbf{S}\mathbf{q_2} - \mathbf{p_2}, \boldsymbol{\nu}_2 \rangle = 0$. Equivalently, by using the $\mathbf{R}$ and $\mathbf{u}$ components of $\mathbf{S}$: $\langle \mathbf{R}\mathbf{q_0} - \mathbf{p_0} + \mathrm{u}, \boldsymbol{\nu}_0 \rangle = 0, \langle \mathbf{R}\mathbf{q_1} - \mathbf{p_1} + \mathrm{u}, \boldsymbol{\nu}_1 \rangle = 0$, and $\langle \mathbf{R}\mathbf{q_2} - \mathbf{p_2} + \mathrm{u}, \boldsymbol{\nu}_2 \rangle = 0$. Since the normals are linearly independent, u can be expressed as $u = \alpha\boldsymbol{\nu}_0 + \beta\boldsymbol{\nu}_1 + \gamma\boldsymbol{\nu}_2$. By substituting this into the constraint equations, the coefficients α, β, γ follow. □

The system of equations defining the manipulation control function $f : \mathcal{B} \times \mathcal{C} \times \mathcal{P}(\mathcal{M}) \rightarrow \mathcal{F}$ has identical structure to the system for the tetrahedron and the same results follow. Thus the instantaneous rotations for polyhedra are also generated by Algorithm 5.1.

5.1.5 One-step Rotations

In the previous sections we have developed an algorithm that gives the instantaneous direction of tracking and the finger forces for rotations of polyhedra. When the desired axis of rotation is specified as a function of time, this approach allows the control of the tracking finger continuously, by following step-wise infinitesimal motions. As the tracking finger moves along a curve on its face, the object moves relative to the fixed fingers.

Notation: P is the polyhedron and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$ the fingers of the hand.

Input: the geometry of the object $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$

the location of the fixed contacts $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$

the constants $\mathbf{I}, m$

current $\mathbf{q_3}, \mathbf{\Omega}, \ddot{\mathrm{u}}, \dot{\mathrm{u}}, \mathrm{u}, f_0, f_1, f_2, f_3$

desired $\dot{\mathbf{\Omega}}$

Repeat:

Instantaneous finger tracking according to Algorithm 5.1.

Update $\mathbf{q_3}, \mathbf{\Omega}, \ddot{\mathrm{u}}, \dot{\mathrm{u}}, \mathrm{u}, f_0, f_1, f_2, f_3$

until the termination predicate $\lambda(\mathbf{p_0}, \ldots \mathbf{p_n}, \mathbf{f_0}, \ldots, \mathbf{f_3})$.

Figure 5.2: Algorithm 5.2 for One-step Finger Tracking

Consider Algorithm 5.2 for One-step tracking given in Figure 5.2. The body of the loop comprises Algorithm 5.1 for instantaneous finger tracking. The termination predicate $\lambda(p_0, \dots p_n, \mathbf{f_0}, \dots, \mathbf{f_3})$ comprises the conditions for all the required finger forces to be valid, and all the finger contacts to be in the interior of the polyhedral faces. The process cannot continue *ad infinitum*, since after a while, one of the grasping fingers will reach an edge and break contact. While it is clear that the motion of the finger can be controlled continuously over time by repeatingly solving the linear system of Theorem 5.1.6, we have to consider what happens when a grasping finger is about to fall off its face. The first step in addressing this problem is to detect when an exceptional event is about to occur. The solution of the system of Theorem 5.1.6 may be used to determine the new contact locations after a small lapse of time. These new locations can be checked against the face boundaries. As long as they are in the interior of the face boundary, the finger tracking process can continue. Concretely, we can augment the system of Theorem 5.1.6 with the the inequalities defining the faces of the polyhedron. The system is then solved subject to the constraints of those inequalities.

In practice, it is not sufficient just to detect when an edge is reached. A grasp with a finger close to an edge is a weak grasp. Consequently, it is desirable to keep all the fingers close to the center of their respective faces. To do this, define the *free region* of a finger on a face to be a polygon similar to the face, where the similarity ratio is $\frac{1}{1+\epsilon}$, $\epsilon > 0$. Assuming uniform mass distribution, the free region has the same center of gravity and orientation as the original face. We ask that the fingers stay within their free region by augmenting the system of motion with the inequality constraints defining the free region. This condition imposes that at all time, the fingers are at least some ϵ distance away from the boundary, where ϵ is a parameter of the problem and depends mostly on the uncertainty model.

Of particular interest is the case when the desired axis of rotation is constant over time. This corresponds to maintaining planar rotations of the body.

Corollary 5.1.12 *The system of Theorem 5.1.6 can be used to generate and maintain planar rotational motions for the grasped object.*

Corollary 5.1.13 *Given an initial orientation R_i of the polyhedron, and a goal orientation R_f, the planar path in M that takes the polyhedron from start to goal is $R_i R_f^{-1}$.*

Proof: $R_i R_f^{-1}$ is the rotation by θ about axis u. The path in $SO(3)$ from start to goal is given by $\gamma(t) = R_{u,\omega t} R_i$. Note that $R_{u,0} = I$ so $\gamma(0) = IR_i = R_i$. Note that $R_{u,\omega t} = R_{u,\theta} = R_f R_i^{-1}$, so $\gamma(t) = R_f R_i^{-1} R_i = R_f$.

□

Planar rotational motions represent an important class of primitives for high-level, task directed applications. Every reorientation can be expressed as the composition of planar rotational motions. Our algorithm allows for a simple control strategy of planar motions, by commanding sliding motions for the tracking finger. This type of operation would clearly relieve the robot programmer from the complexity inherent in low-level manipulation control.

5.1.6 Multi-step Rotations

When one of the grasping fingers attempts to move past the boundary of its free region, or when the matrix of the linear system is singular, the tracking finger cannot continue its action. However, the rotation process may continue by replacing the guilty finger. Depending on what causes the singularity of the process, three cases may occur.

1. The matrix is singular. Then there is no direction for the tracking finger that would generate the desired rotation. A free finger must replace the old tracking finger. The challenge is to find an initial position on the surface of the object for the new tracking finger that would yield the desired rotation for the object.

2. The tracking finger has reached the boundary of its free region. A new tracking finger must be assigned to participate in the grasp of the object

with the fixed fingers. The challenge is again to find an initial position on the surface of the object for the new tracking finger that would give the same desired axis of rotation.

3. A fixed finger has reached a boundary. Then the previously tracking finger becomes a fixed finger, thus changing the fixed finger set. The new fixed finger set defines different constraints and a different configuration space. The technique outlined in Section 5.1.2 is used to determine the relevant equations. What is left then is to find a new face and a starting point on the new tracking finger that would produce the desired acceleration on the body.

The common theme of all these extreme cases is that a new tracking finger must be found, as well as its initial location that will keep the rotation process continuous. A preliminary condition for the replacement is that the free finger set of the robot hand be not empty. It is not always possible to find a point on the object that satisfies this property. All the faces of the object may be inspected for the feasibility of a starting location for the tracking finger.

Corollary 5.1.14 *Given a new face, and a desired axis of rotation, the starting location of the new tracking finger on that face, if it exists, can be derived from the system of Theorem 5.1.6.*

Figure 5.3 shows the multi-step algorithm. In the preprocessing stage, a graph whose vertices are all tuples of faces on which the polyhedron can me grasped. The edges represent transitions between tuples of faces that differ only by one. This graph is used to decide the face for the new tracking finger for the cases that require a change of face to continue the process.

5.1.7 The Effect of Gravitation

The motion system does not have gravitation explicit in the equations. This omission is appropriate for applications in which the forces exerted by the fingers are

Notation: P is the polyhedron and $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}, \mathbf{f_4}$ the fingers of the hand.

t is the index of the tracking finger and l the index of the free finger.

$S = (V, E)$ is the transition graph.

$V = \{(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j, \boldsymbol{\nu}_k, \boldsymbol{\nu}_l) | \exists$ grasp on the respective faces $\}$

$E = \{(v_1, v_2) | v_1, v_2 \in V, v_1 = (\boldsymbol{\nu}_i, \boldsymbol{\nu}_j, \boldsymbol{\nu}_k, \boldsymbol{\nu}_l), v_2 = (\boldsymbol{\nu}_j, \boldsymbol{\nu}_k, \boldsymbol{\nu}_l, \boldsymbol{\nu}_m)\}$.

Input: the geometry of the object $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$

the location of the fixed contacts $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$

the constants $\mathbf{I}, m$

current $\mathbf{q_3}, \boldsymbol{\Omega}, \ddot{\mathrm{u}}, \dot{\mathrm{u}}, \mathrm{u}, f_0, f_1, f_2, f_3$

desired $\dot{\boldsymbol{\Omega}}$

Preprocessing: Build S.

Initialization: Let $v \in V$.

Assign $\mathbf{f_0}$ to first(v), $\mathbf{f_1}$ to second(v), $\mathbf{f_2}$ to third(v), $\mathbf{f_3}$ to fourth(v).

Grasp P using $\mathbf{f_0}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$.

$r = 0; t = 0; n = 0; l = 4$

Repeat:

One-step tracking with $\mathbf{f_t}$ according to Algorithm 5.2.

$n = n + 1$; $t = l$; $l = (l + 1) \bmod 5$.

$v =$ select-adjacent(v).

Calculate the location and force for $\mathbf{f_t}$ according to Theorem 5.1.6.

Figure 5.3: Algorithm 5.3 for Multi-step Finger Tracking

large, compared to the gravitational forces. It also makes this theory applicable to gravitationless environments, for problems related to the exploration of space. It is conceivable to imagine a task-directed application in which most of the work is done by gravity, and the fingers do only some guiding. For these kinds of applications, it is important to include explicitly the terms that arise due to gravitation. For the effect of a uniform gravitational field on a system of particles, in an inertial frame, gravity acts as if all the force is applied through the center of mass. In an inertial frame, gravity is also an external force. For the equations relative to the moving object-fixed frame, it is easy to see that to include the effect of gravitation, the total external force gets an extra factor of Mg. In an inertial system, if the origin of the frame of reference is not the center of mass, then the external torque gets an extra factor of $m\mathbf{g} \times \mathbf{r}$. The external torque due to gravity is zero if the center of mass is the center of the coordinate system, because $\mathbf{N_g} = \sum m_i \mathbf{g} \times \mathbf{r_i} = \mathbf{g} \times (\sum m_i \mathbf{r_i})$. The object-fixed system is not inertial, but by the same argument, the external torque gets a factor of $m\mathbf{r} \times \mathbf{g} - \mathbf{u}_t \times m\mathbf{g}$.

5.2 Rotations of Polyhedra with Hardfinger Contacts

For many applications, the frictional component of the interaction between two objects can not be ignored. A hardfinger contact models point contacts in the presence of friction. Friction enlarges the range of reaction forces that can be produced by an object. We model friction with Coulomb's law, which represents the range of possible reaction forces by a cone called *friction cone.* The axis of this cone is the normal to the surface of the object at the point of contact and its apex angle is $\theta = \tan^{-1}\mu, 0 \leq \mu \leq 2\frac{\pi}{2}$ where μ is the *coefficient of friction.* Any force directed inside the friction cone is a possible reaction force.

To model the effect of friction between the fingers and the object, assume a uniform coefficient f friction μ. Let $\mathbf{f_i}$ be the force exerted by finger f_i. This force is no longer normal, but it can be decomposed into a normal component and a tangential component: $\mathbf{f_i} = \mathbf{f_i^n} + \mathbf{f_i^t}$. The new forces $\mathbf{f_i^n}$ and $\mathbf{f_i^t}$ are two new unknows for the system of motion. To determine these forces, new equations are added to the system of motion [Goy89]. If the relative velocity at the contact between $\mathbf{f_i}$ and $\mathbf{q_i}$ is not zero, that is $\dot{\mathbf{q}}_\mathbf{i} \neq \mathbf{0}$, then the finger is slipping and the equations:

$$\mathbf{f_i^t} = -\mu\|\mathbf{f_i^n}\|\frac{\dot{\mathbf{q}}_\mathbf{i}^\mathbf{t}}{\|\dot{\mathbf{q}}_\mathbf{i}^\mathbf{t}\|}. \tag{5.16}$$

$$\mathbf{f_i} = \mathbf{f_i^t} + \mathbf{f_i^n}. \tag{5.17}$$

If the relative velocity at the contact between $\mathbf{f_i}$ and $\mathbf{q_i}$ is zero, then the finger is sticking and depending on the value of the relative tangential acceleration of the point of contact, different equations are added. When the relative tangential acceleration $\ddot{\mathbf{q}}_\mathbf{i} = \mathbf{0}$, then the behavior at the contact point is continued sticking and the extra equations are:

$$\ddot{\mathbf{q}}_\mathbf{i} = 0. \tag{5.18}$$

$$\|\mathbf{f_i^t}\| \leq \mu\|\mathbf{f_i^n}\|. \tag{5.19}$$

When the relative tangential acceleration at the contact is not zero, there is impending slip [Goy89] and the added equation is

$$\mathbf{f_i^t} = -\mu\|\mathbf{f_i^n}\|\frac{\ddot{\mathbf{q}}_\mathbf{i}^\mathbf{t}}{\|\ddot{\mathbf{q}}_\mathbf{i}^\mathbf{t}\|}. \tag{5.20}$$

In all these cases, the system of motion can be solved as in Theorem 5.1.6.

5.3 Rotations of Objects with Arbitrary Surfaces

A different instantiation for the manipulation system $\mathcal{S}$ can be done for the class of objects with arbitrary surfaces.

1. $\mathcal{B}$ is the set of objects with arbitrary surfaces. Let B be an element of this set.

2. $\mathcal{C}$ is the set of constraints that expresses a continuous four finger grasp on the object. In addition there are three kinematic constraints that describe the fact that at all time, three of the robot fingers stay fixed in space. Moreover, the manipulation is being done under the laws of Newtonian mechanics. Let C be an element of this set.

3. $\mathcal{F}$ is the set of robot hands with four fingers. Each finger has a limitless reach region, and the finger tips are frictionless points. Let F be an element of this set.

4. $\mathcal{M}$ is the configuration space defined by these sets. Let M be an element of this set.

For this instantiation, each $B \in \mathcal{B}$ is given in terms of its faces which are given as implicit surfaces $s(x, y, z) = 0$. The problem is to define the control function that accomplishes the reorientation of the object under the desired angular acceleration $\dot{\mathbf{\Omega}}$.

Theorem 5.3.1 *Let $s_i : R^3 \rightarrow R, i = 0, 1, 2$ be the functions defining the three surfaces for the fixed fingers. The acceleration constraints that arise from constraining the fixed fingers to keep continuous contact with these fingers are:*

$$(\mathbf{\Omega} \times (\mathbf{q}_{i,t} - \mathbf{u}) + \dot{\mathbf{u}}) Hess(s_i)(\mathbf{\Omega} \times (\mathbf{q}_{i,t} - \mathbf{u}) + \dot{\mathbf{u}})^T + \langle \nabla s_i, (\dot{\mathbf{\Omega}} \times (\mathbf{q}_{i,t} - \mathbf{u}) + \mathbf{\Omega} \times (\mathbf{\Omega} \times (\mathbf{q}_{i,t} - \mathbf{u})) + \ddot{\mathbf{u}} \rangle = 0.$$

Proof: Let $\mathbf{q}_i(t), i = 0, 1, 2$ be the locations of the three fixed fingers on their surfaces over time. Then the constraint equations are $s_i(\mathbf{q}_i(t)) = 0, i = 0, 1, 2$.

By applying the same observations as in the case of polyhedra, we can express the velocity and acceleration of the contact points at arbitrary time. Consider the relationship between vertex $\mathbf{q}$ at time 0, also denoted by $\mathbf{q}_0$, and vertex $\mathbf{q}$ at time t, denoted by $\mathbf{q}_t$. For the velocity, we have that $\mathbf{q}_t = \mathbf{R}_t \mathbf{q}_0 + \mathbf{u}_t$, so $\mathbf{q}_0 = \mathbf{R}_t^{-1}(\mathbf{q}_t - \mathbf{u}_t)$. By differentiating we have that $\dot{\mathbf{q}}_t = \dot{\mathbf{R}}_t \mathbf{q}_0 + \dot{\mathbf{u}}_t$ and by replacing the value for $\mathbf{q}_0$ it follows that $\dot{\mathbf{q}}_0 = \dot{\mathbf{R}}_t\mathbf{R}_t^{-1}(\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t$. $\dot{\mathbf{R}}_t\mathbf{R}_t^{-1}$ is a skew symmetric matrix which is an element of the Lie algebra of the rotation group $\mathbf{SO}(3)$. Its property is that for a vector $\mathbf{v}$, $\mathbf{Y}_t\mathbf{v} = \boldsymbol{\Omega} \times \mathbf{v}$, where $\boldsymbol{\Omega}$ is the vector defined by the three non zero values of $\mathbf{Y}_t$. So this becomes $\dot{\mathbf{q}}_t = \boldsymbol{\Omega}_t \times (\mathbf{q}_t - u_t) + \dot{\mathbf{u}}_t$. This implies further that $\ddot{\mathbf{u}}_t = \dot{\boldsymbol{\Omega}}_t \times (\mathbf{q}_t - \mathbf{u}_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q}_t - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t$.

Now we can see that $\frac{d}{dt}s_i = \langle \nabla s_i, \dot{\mathbf{q}}_t \rangle = 0$, or $\langle \nabla s_i, \boldsymbol{\Omega}_t \times (\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t \rangle = 0$. For the acceleration constraint, we have that $\frac{d^2}{dt^2}s_i = \dot{\mathbf{q}}_t \mathrm{Hess}(s_i)\dot{\mathbf{q}}_t^T + \langle \nabla s_i, \ddot{\mathbf{q}}_t \rangle = 0$. By replacing the terms related to $\mathbf{q}_0$, this constraint becomes $(\boldsymbol{\Omega}_t \times (\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t)\mathrm{Hess}(s_i)(\boldsymbol{\Omega}_t \times (\mathbf{q}_t - \mathbf{u}_t) + \dot{\mathbf{u}}_t)^T + \langle \nabla s_i, (\dot{\boldsymbol{\Omega}}_t \times (\mathbf{q}_t - \mathbf{u}_t) + \boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{q}_t - \mathbf{u}_t)) + \ddot{\mathbf{u}}_t \rangle = 0$.
□

Thus, the set of equations defining the control for this problem is: is:

$$(\boldsymbol{\Omega} \times (\mathbf{q}_{0,t} - \mathbf{u}) + \dot{\mathbf{u}})\mathrm{Hess}(s_0)(\boldsymbol{\Omega} \times (\mathbf{q}_{0,t} - \mathbf{u}) + \dot{\mathbf{u}})^T + \langle \nabla s_0, (\dot{\boldsymbol{\Omega}} \times (\mathbf{q}_{0,t} - \mathbf{u}) + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times (\mathbf{q}_{0,t} - \mathbf{u})) + \ddot{\mathbf{u}} \rangle = 0. \tag{5.21}$$

$$(\boldsymbol{\Omega} \times (\mathbf{q}_{1,t} - \mathbf{u}) + \dot{\mathbf{u}})\mathrm{Hess}(s_1)(\boldsymbol{\Omega} \times (\mathbf{q}_{1,t} - \mathbf{u}) + \dot{\mathbf{u}})^T + \langle \nabla s_1, (\dot{\boldsymbol{\Omega}} \times (\mathbf{q}_{i,t} - \mathbf{u}) + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times (\mathbf{q}_{1,t} - \mathbf{u})) + \ddot{\mathbf{u}} \rangle = 0. \tag{5.22}$$

$$(\boldsymbol{\Omega} \times (\mathbf{q}_{2,t} - \mathbf{u}) + \dot{\mathbf{u}})\mathrm{Hess}(s_2)(\boldsymbol{\Omega} \times (\mathbf{q}_{2,t} - \mathbf{u}) + \dot{\mathbf{u}})^T + \langle \nabla s_2, (\dot{\boldsymbol{\Omega}} \times (\mathbf{q}_{2,t} - \mathbf{u}) + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times (\mathbf{q}_{2,t} - \mathbf{u})) + \ddot{\mathbf{u}} \rangle = 0. \tag{5.23}$$

$$s_3(\mathbf{q}_3(t)) = 0. \tag{5.24}$$

$$(\boldsymbol{\Omega}_t \times (\boldsymbol{\Omega}_t \times (\mathbf{r} - \mathbf{u}_t)) - \dot{\boldsymbol{\Omega}}_t \times (\mathbf{r} - \mathbf{u}_t) + 2\Omega_t \times \dot{\mathbf{u}}_t - \ddot{\mathbf{u}}_t = \frac{1}{m}\sum f_i \boldsymbol{\nu}_i. \tag{5.25}$$

$$\mathbf{I}\dot{\boldsymbol{\Omega}} + m(\dot{\mathbf{u}}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t) + \mathbf{u}_t \times (\dot{\boldsymbol{\Omega}} \times \mathbf{u}_t) + \mathbf{u}_t \times (\boldsymbol{\Omega} \times \dot{\mathbf{u}}_t)) = (\mathbf{I}\boldsymbol{\Omega}) \times \boldsymbol{\Omega} + m(\mathbf{u}_t \times (\boldsymbol{\Omega} \times \mathbf{u}_t)) \times \boldsymbol{\Omega} + \sum_{i=0}^{3} f_i \mathbf{q}_i \times \boldsymbol{\nu}_i - \sum_{i=0}^{3} \mathbf{u}_t \times f_i \boldsymbol{\nu}_i. \tag{5.26}$$

Theorem 5.3.2 *Given an object whose faces are specified by implicit surfaces, a robot hand with four fingers operating with three constraining fingers and one* *tracking finger, and a curve γ in the configuration space, at any point in time the tracking motion required to generate γ is given by a generally solvable linear system.*

Proof: Consider the motion system as outlined above. To solve the system for the tracking finger $\mathbf{q_3}$ we note that $\mathbf{q_3}$ appears linearly only due to the force and torque equations. Thus, the solution for the general system can be obtained in a manner which is very similar algebraically to the solution of the system for polyhedra. □

5.4 Summary

In this chapter we have analyzed in detail the dynamics of planning manipulation strategies by finger tracking for objects whose surfaces can be specified as time-varying functions, that are being reoriented by robot hands with frictionless contacts, where the reorientation is specified as a time-varying function. The instantaneous finger forces and tracking location are determined by a linear system. This algorithm can be used to maintain planar rotations. We have argued that planar rotations are good high-level primitives for task-directed planning.

Chapter 6

Simulation

The first phase for the validation of the constraint-based manipulation in R^3 by finger tracking was done in simulation. Simulation systems are well suited for the analysis of intricate problems such as the manipulation problem, for they provide data that can suggest directions in understanding properties that would otherwise be difficult to detect. Simulation systems can formulate and integrate a complex set of equations describing the behavior of complex objects [HH86,Cre89]. To this effect, they allow easy specifications of geometry and dynamics, they support a wide range of analyses, and they enable the use of these analyses for design modification. We use simulation to verify that the observed behavior of the finger tracking control matches the theoretical predictions. We also use the numerical data generated during simulation for guidance towards further insights into the structure of the problem. In particular, this data was used as a starting point for the singularity and feasibility analysis of the motion system. The first phase is being followed by experimentation with mobile robots for large scale manipulation.

6.1 The Simulation of Manipulation

A special purpose dynamic simulation system was built for the finger tracking algorithm. This system consists of the object being manipulated, the hand that does the manipulation, and the constraints for the interaction. The quantities of the simulations are modelled as collections of rigid bodies related by constraints. Constrained relationships define the interaction between the hand and the object. To carry out a simulation, the motion equations from each rigid body are collected and assembled into a system of equations which can be solved to determine the instantaneous accelerations and hinge control forces. The derived accelerations are integrated to produce the motion of the simulated object.

Figure 6.1 describes the main simulation loop. For simulation, the dynamic module needs a variety of geometry information, which includes the geometry and dynamical properties of the objects of the system, and the equations of motion. Of particular importance is the data structure called the *state vector*, which comprises the position, orientation in Euler parameters, velocity, angular velocity, acceleration, and angular acceleration of the objects, all specified in global coordinates. The current state of the system is an instantiation of the state vector variables for the current time.

The integration phase generates new positions and velocities from their old positions, velocities, and accelerations. Several integration methods are available, such as Adams-Moulton, Runge-Kutta, and the adaptive time-step method, by interfacing with the Newton simulator [Cre89].

Build-system is a module that sets up the manipulation linear system and solves for the unknown variables. The equations are build following the proof of Theorem 5.1.6, and then solved for the position of the tracking finger and the forces exerted by the four grasping fingers.

Special-events is a module that detects whether any finger is about to lose contact with the object, or whether any of the forces required to maintain the motion is outside the valid range of values. The first aspect is handled by geometric analysis, and the second by a numerical analysis of the motion system. When such a special event is encountered, this module is responsible for optimally choosing a different tracking finger.

Initialize: system variables and state.

Repeat: Save current state.

Integrate from current state.

Build system of equations for motion.

Handle exceptional events.

Convert.

Report.

until stop simulation condition.

Figure 6.1: Algorithm 6.1 for Simulation

Convert is a module that allows the conversion of the position, orientation, velocity, and acceleration data between the object-fixed system of coordinates and the hand-fixed system of coordinates, and between the object-fixed formulation of the problem and the format of the state vector used by the integrator. The effective difference between the outcome of computation carried in the hand-fixed system and computation carried in the object-fixed system is only in the viewpoint of the resulting motion. In a hand-fixed coordinate system, the computation generates the motion of the object relative to the fixed fingers. In an object-fixed system, the computation generates the motion of the four fingers relative to the faces on which they act. These two motions are inverses of one another, and are related by the transformation matrices described in Chapter 5.

Report is a problem-specific analysis and visualization package, that allows reasoning about the simulation in the object-fixed system of coordinates simultaneously with reasoning in the hand-fixed system of coordinates.

6.1.1 The Data

The object being manipulated is a polyhedron, which is specified geometrically completely in terms of its vertices $\mathbf{p_0}, \ldots, \mathbf{p_n}$; the location of the center of mass, its mass, and its inertia tensor. The robot hand is described in terms of its fingers, which are represented in a simplified way as sphperes of a certain radius. The initial locations of the fingers on the surface of the object $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$ and the forces exerted by the fingers f_0, f_1, f_2, f_3 are also known.

The state variable consists of position, orientation, velocity, angular velocity, acceleration, and angular acceleration for the object being manipulate: $\mathbf{u}, \mathbf{R}, \dot{\mathbf{u}}$, and $\mathbf{\Omega}$.

6.1.2 The Algorithm for Simulation

The initialization stage consists of assigning values to initialize the geometric and dynamic data about the polyhedron and the hand, as well as their initial kinematic interaction. These quantities are described in $\hat{x}$, the coordinate system fixed on the object. This choice for the system of coordinates keeps the vertices $\mathbf{p_0}, \ldots, \mathbf{p_n}$ constant throughout the simulation.

In general, at time t, the known quantities are $\mathbf{u}_t, \mathbf{R}_t, \dot{\mathbf{u}}_t, \mathbf{\Omega}_t, \mathbf{q_i}$, and f_i, for $i = 0, 1, 2, 3$. To advance in time, the simulator solves the system of equations to determine $\ddot{\mathbf{u}}_t$ and then with a single integration step, it generates $\mathbf{u}_{t+\Delta t}, \mathbf{R}_{t+\Delta t}, \dot{\mathbf{u}}_{t+\Delta t}$, and $\mathbf{\Omega}_{t+\Delta t}$ at time $t + \Delta t$. These new values are used to build the system of Theorem 5.1.6, and then to solve for $\mathbf{q_3}, f_0, f_1, f_2, f_3$ at time $t + \Delta t$. These values are further used to determine the locations for $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$ at time $t + \Delta t$. For each $\mathbf{q_i}$, $i = 0, 1, 2$, $\mathbf{q_i}_{t+\Delta t} = \mathbf{R}_{t+\Delta t}\mathbf{q_i}_0 + \mathbf{u}_{t+\Delta t}$.

The state of the system in object-fixed coordinates at time $t+\Delta t$ is related to the state in hand-fixed coordinates at the same time by $\mathbf{R}$ and $\mathbf{u}$. In particular, the new locations for the vertices of the polyhedron are $\mathbf{p_i}_{t+\Delta t} = \mathbf{R}^{-1}_{t+\Delta t}\mathbf{p_i}_0 - \mathbf{R}^{-1}_{t+\Delta t}\mathbf{u}_{t+\Delta t}$. These values are used for the visualization of the simulation results in the global, hand-fixed frame.

Notation:$\hat{x}$ a coordinate system fixed on the object.

$\hat{y}$ a coordinate system fixed on the robot hand.

Input: desired $\dot{\mathbf{\Omega}}$.

Initialization:the geometry: $\mathbf{p_0}, \ldots, \mathbf{p_n}$ (in $\hat{x}$).

the initial contacts: $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$ (in $\hat{x}$), and f_0, f_1, f_2, f_3.

the state vector: $\mathbf{u}, \mathbf{R}, \dot{\mathbf{u}}, \mathbf{\Omega}, \ddot{\mathbf{u}}$, and $\dot{\mathbf{\Omega}}$ (in $\hat{x}$).

Repeat: Save current state $\mathbf{u}, \mathbf{R}, \dot{\mathbf{u}}$, and $\mathbf{\Omega}$

Use the system of Theorem 5.1.6 to determine new $\mathbf{q_3}, f_0, f_1, f_2, f_3$.

Update $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$

Handle exceptional events.

Transform variables to hand-fixed system $\hat{y}$.

Report (in $\hat{y}$.)

Determine $\ddot{\mathbf{u}}$ from the motion equations

Integrate from current state.

until stop simulation condition.

Figure 6.2: Algorithm 6.2 for Simulation of Manipulation

An exceptional event occurs when the force f_0, f_1, f_2, f_3 are outside their legal range, or when one of the finger contacts reaches an edge. The special event handler detects these conditions and resolves the situation by selecting a new tracking finger and by determining the contact compatible with the desired motion. The function that builds and solves the system of Theorem 5.1.6 is used in conjunction with the graph describing legal grasp transitions.

6.1.3 Conversion to and from Euler Parameters

As pointed out before, the visualization for the simulation of this problem comprises the motion of the polyhedron, from a viewpoint fixed on the robot hand, or the paths of the fingers relative to the polyhedral faces, from a viewpoint fixed on the object. Since both viewpoints of the motion are useful, the module Convert was introduced to allow for alternation between them.

An important component of this conversion is the representation of the rotational and translational components of the paths. The state variable uses a vector representation for the translation, and a vector representation based on Euler parameters for rotations. In order to use the rotational data produced by the simulation to generate the appropriate paths, the conversion between the the representation of rotations by Euler parameters and the representations of rotations as matrices is needed.

Let e_0, e_1, e_2, e_3 be the Euler parameters representation of a rotation and $\mathbf{R}$ the usual matricial representation. The relation between the two representations is given by [Wit77]:

$$\mathbf{R} = \begin{pmatrix} 2(e_0^2 + e_1^2) - 1 & 2(e_1 e_2 + e_0 e_3) & 2(e_1 e_3 - e_0 e_2) \\ 2(e_1 e_2 - e_0 e_3) & 2(e_0^2 + e_2^2) - 1 & 2(e_2 e_3 + e_0 e_1) \\ 2(e_1 e_3 + e_0 e_2) & 2(e_2 e_3 - e_0 e_1) & 2(e_0^2 + e_3^2) - 1 \end{pmatrix}.$$

Given the Euler parameter representation, the rotation matrix can be built as above. Given the rotation, determining the Euler parameters requires some computation. After some simple algebraic manipulation, it follows that $e_0^2 = \frac{tr(\mathbf{R})+1}{4}$ and $e_i^2 = \frac{\mathbf{R}(i,i)}{2} - \frac{tr(\mathbf{R})-1}{4}$, for $i = 1, 2, 3$, where $tr(\mathbf{R})$ is the trace of $\mathbf{R}$. The signs of the Euler parameters can be determined by comparing the values of $\mathbf{R}(i,j) + \mathbf{R}(j,i)$ and $\mathbf{R}(i,j) - \mathbf{R}(j,i)$ for $i \neq j$ against the sixteen sign combinations.

6.2 A Simulation Example

Among the many objects on which the reorientation algorithm was tested, we choose as canonical example the regular tetrahedron. More specifically, the data for the simulation program consists of:

1. Geometry: a regular tetrahedron whose vertices are:

   $\mathbf{p_0} = (0, 0, \frac{2\sqrt{6}}{3}), \mathbf{p_1} = (-1, -\frac{2\sqrt{3}}{6}, 0), \mathbf{p_2} = (1, -\frac{2\sqrt{3}}{6}, 0), \mathbf{p_3} = (0, \frac{2\sqrt{3}}{3}, 0).$

2. Finger contacts described in terms of barycentric coordinates (the notation convention is that finger $\mathbf{f_i}$ is on the face opposing vertex $\mathbf{p_i}$:

   $\mathbf{q_0} = (0.25, 0.25), \mathbf{q_1} = (0.30, 0.30), \mathbf{q_2} = (0.40, 0.40).$

3. The object starts from rest, thus the linear and angular velocities, as well as the linear and angular accelerations, are zero.

4. The desired axis of rotation: $\dot{\mathbf{\Omega}} = (0.4, 0.4, 0.8).$

5. The duration of the simulation: 10 seconds.

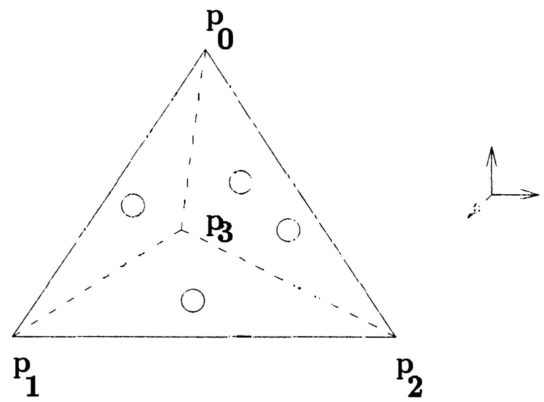

Figure 6.3: The object being simulated

6.2.1 Simulation Results

The simulation is carried out in two stages. In the first phase the *special events* package is disabled. This results in a *one-step* tracking simulation, where only one finger does the tracking. By disabling the detection of special events, the finger forces are allowed to take any values and the fingers are allowed to act on the entire plane supporting their tetrahedral face. The dependency of the force and contact behavior on various parameters is complex and has not been fully analyzed thus far. This dependency is necessary in order to evaluate the optimality of an initial grasp for a given task. By doing simulations that do not consider the thresholds for the unknown variables, the general pattern for the variation of the forces and finger locations can be observed. In the second phase, the *special events* package detects the end of a *one-step* tracking and selects a new tracking finger.

Numerical data from the first phase of the simulation is presented in the form of graphs. The graphs have been generated with *DSTOOL*, a dynamical systems package developed at Cornell [BGM+92].

Figure 6.4, Figure 6.5, and Figure 6.6 represent the curve for the force magnitudes of the fixed fingers, which is determined at each time step by solving the linear system of Theorem 5.1.6. The plotted range is the interval (-100, 100). Figure 6.7 represents the curve for the force magnitude of the tracking finger, again determined by solving the system of Theorem 5.1.6.

Figures 6.8 through Figure 6.13 represent the curve for the relative motion of fixed fingers $\mathbf{q_0}, \mathbf{q_1}$, and q_2 on their respective tetrahedral faces. Since the finger contacts are represented in terms of barycentric coordinates, the plotted variables are the barycentric coordinates, whose range is in $(0, 1)$. The values were determined as $\mathbf{q_{i_t}} = \mathbf{R_t q_i} + \mathbf{u_t}$, for $i = 0, 1, 2$.

Figure 6.14 1nd Figure 6.15 represent the curve for the relative motion of tracking finger $\mathbf{q_3}$ on the tetrahedral face opposing vertex $\mathbf{p_3}$. Since the finger contacts are represented in terms of barycentric coordinates, this graph represents the dependency of the second and third barycentric coordinates of $\mathbf{q_3}$ whose range is in (0, 1), on time. The values were determined by solving the system of Theorem 5.1.6.

There are several noteworthy properties. The four finger forces have similar curves. In particular, the four forces stay within their legal thresholds for approximately the same time interval, an observation relevant to the feasibility of the algorithm. The exact dependency between the force values and the rest of the quantities in the simulation has not been expressed in a closed form so far, but these graphs give insights into what the dependency might be. Another important observation is that the tracking finger goes through a singularity only when the force f_3 changes sign. This can be verified equationally in the initial system of equations describing the constrained motion. For practical applications that involve real robots, the consequence is that the singularity in the tracking finger location is not a concern, since the forces exerted by the fingers are always positive. Finally, we observe that the path of the tracking finger is longer than the paths of the fixed fingers. This can be used as a criterion for choosing a tracking face for objects whose geometry gives rise to several possibilities.

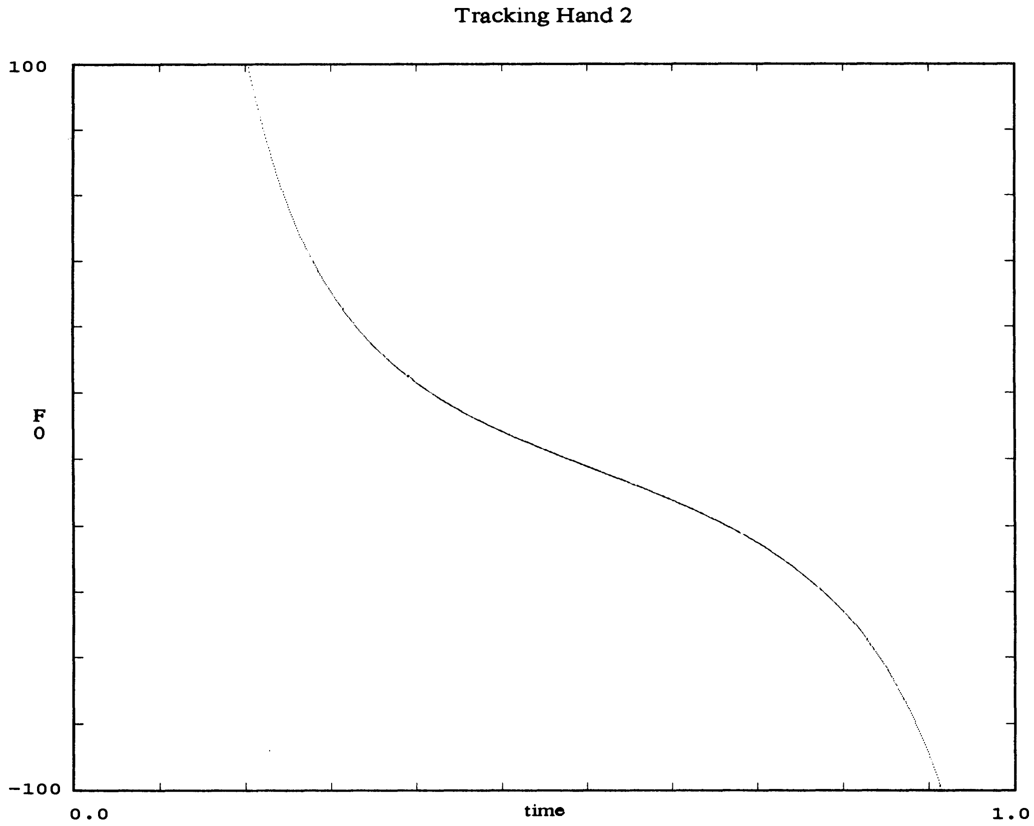

Figure 6.4: The magnitude of the force $\mathbf{f_0}$

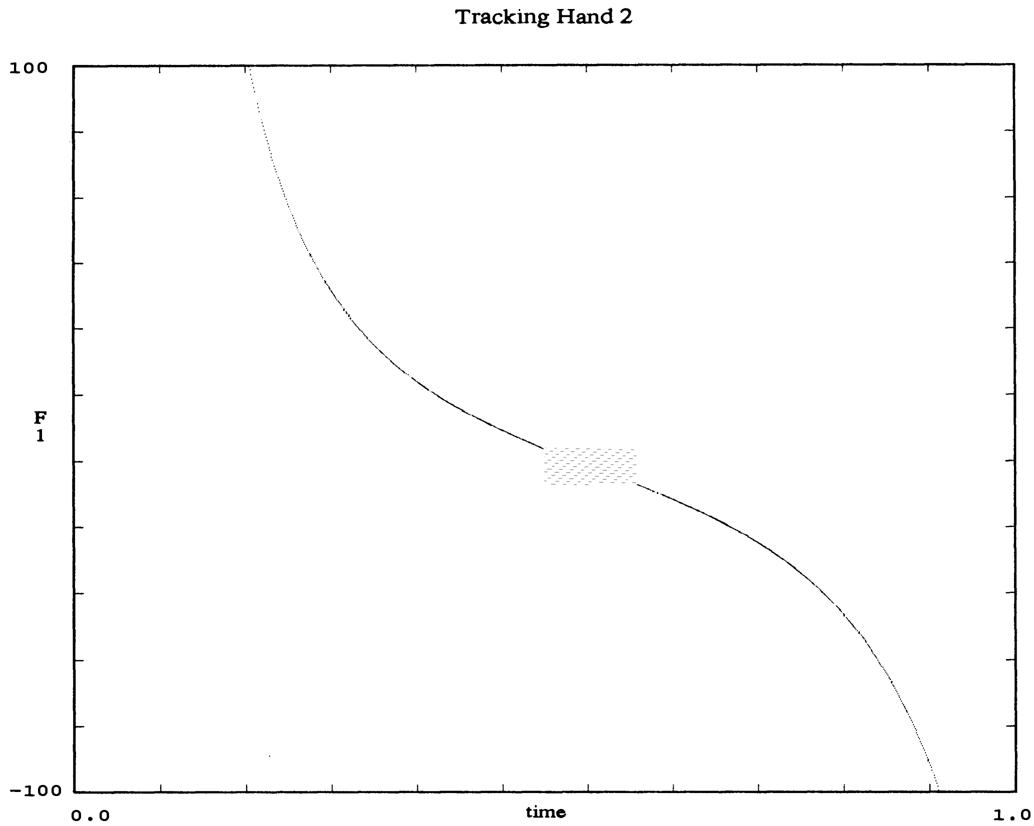

Figure 6.5: The magnitude of the force $\mathbf{f_1}$

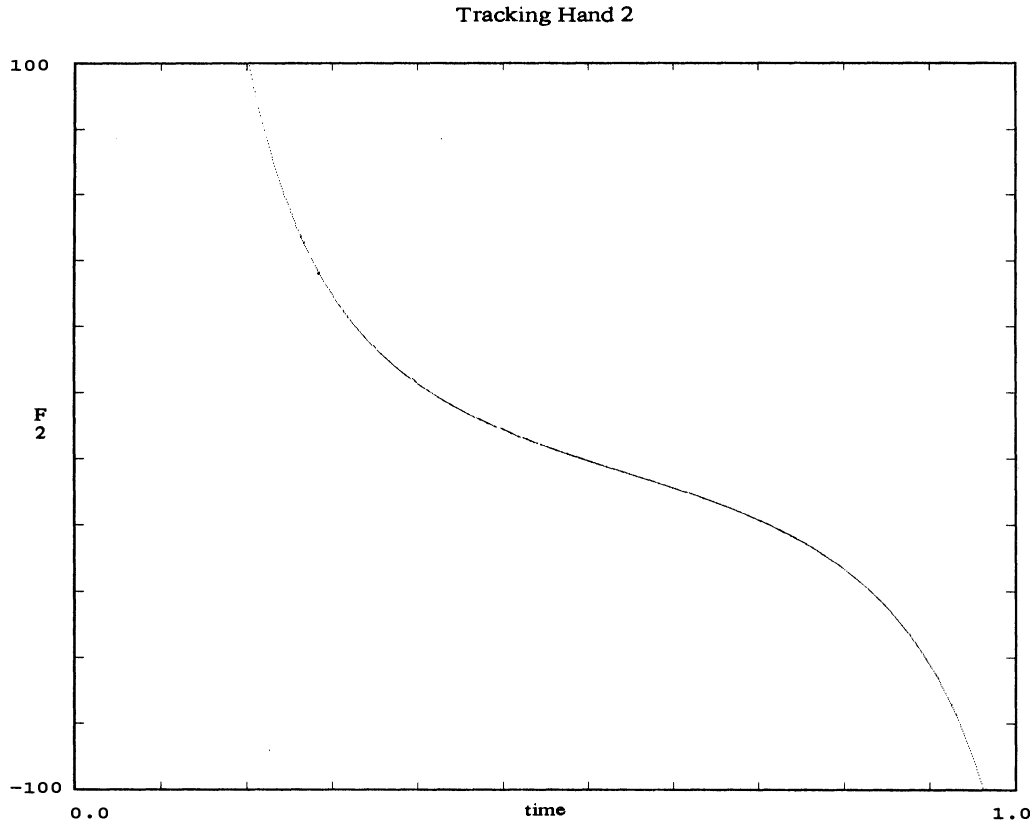

Figure 6.6: The magnitude of the force $\mathbf{f_2}$

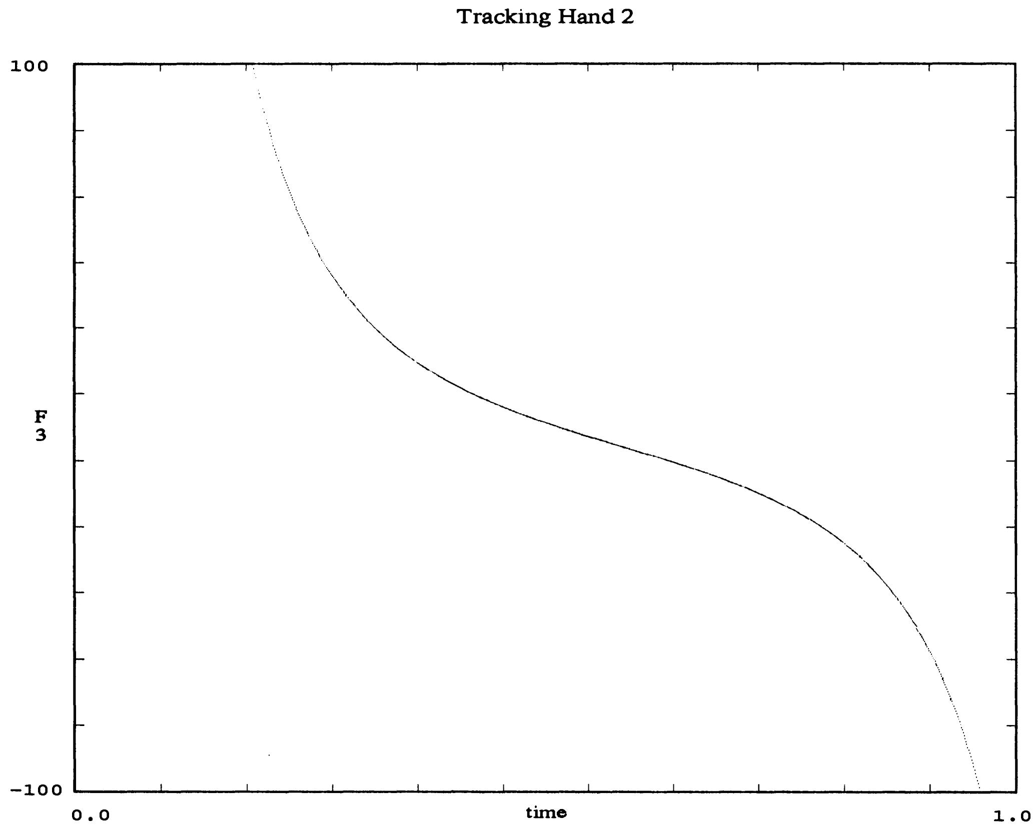

Figure 6.7: The magnitude of the force $\mathbf{f_3}$

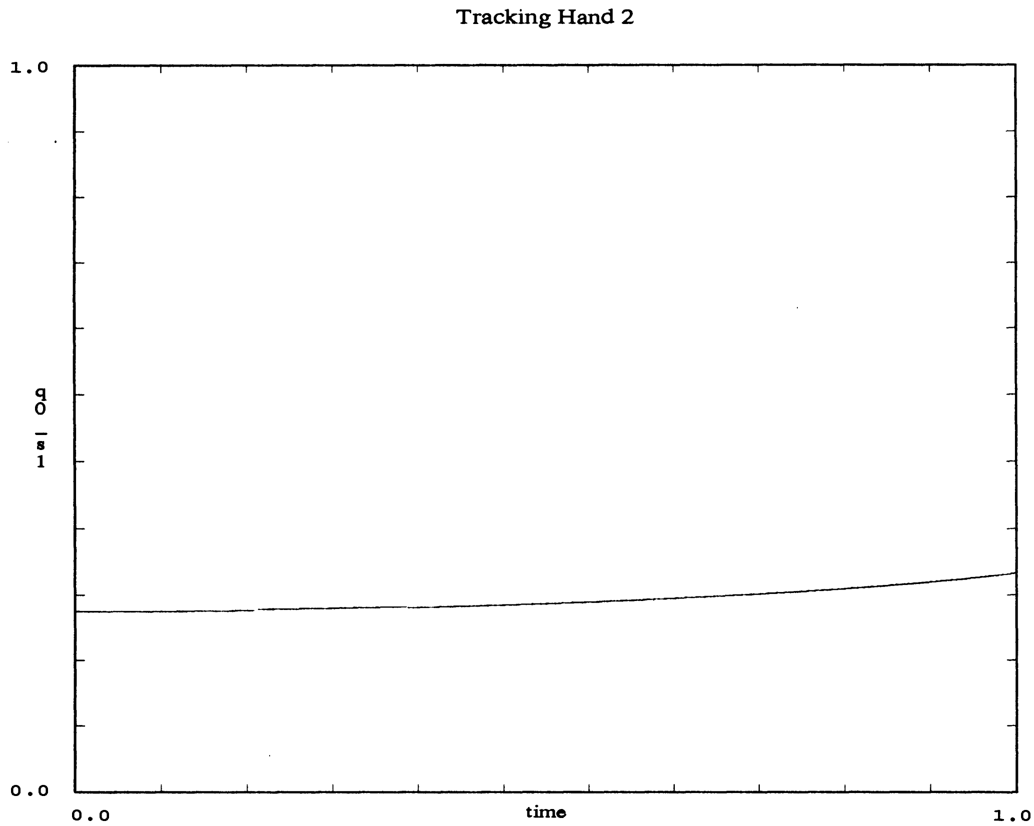

Figure 6.8: The first barycentric coordinate of $\mathbf{q_0}$

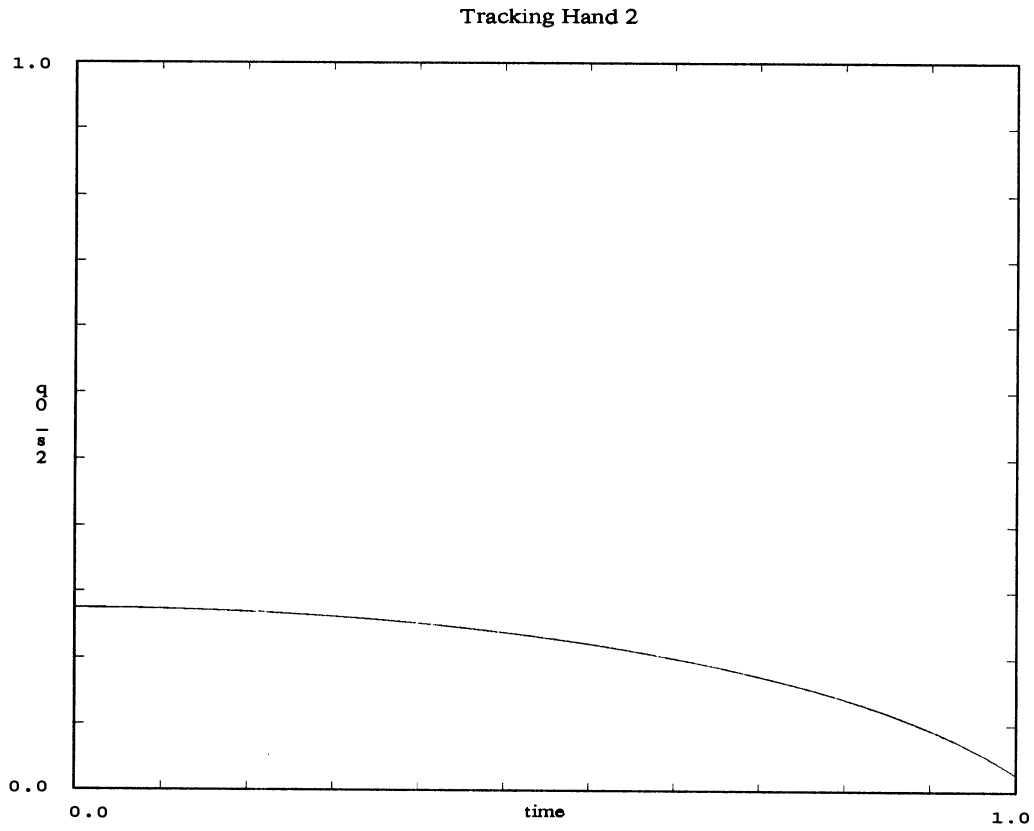

Figure 6.9: The second barycentric coordinate of $\mathbf{q_0}$

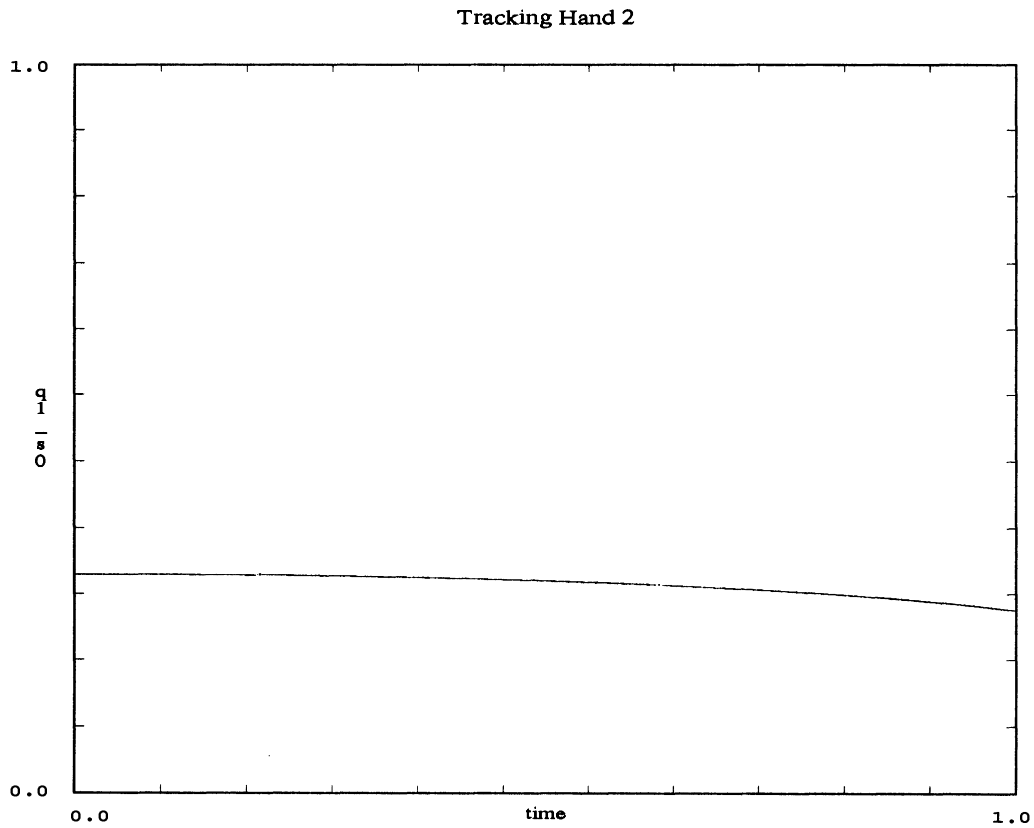

Figure 6.10: The first barycentric coordinate of $\mathbf{q_1}$

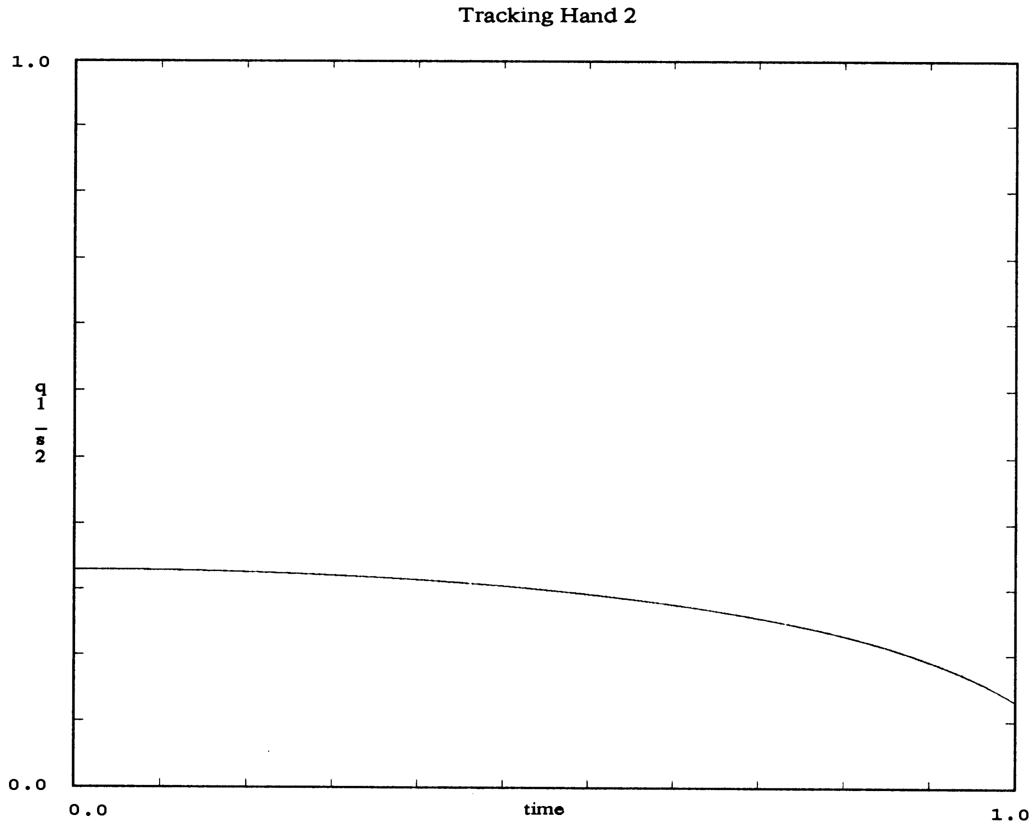

Figure 6.11: The second barycentric coordinate of $\mathbf{q}_1$

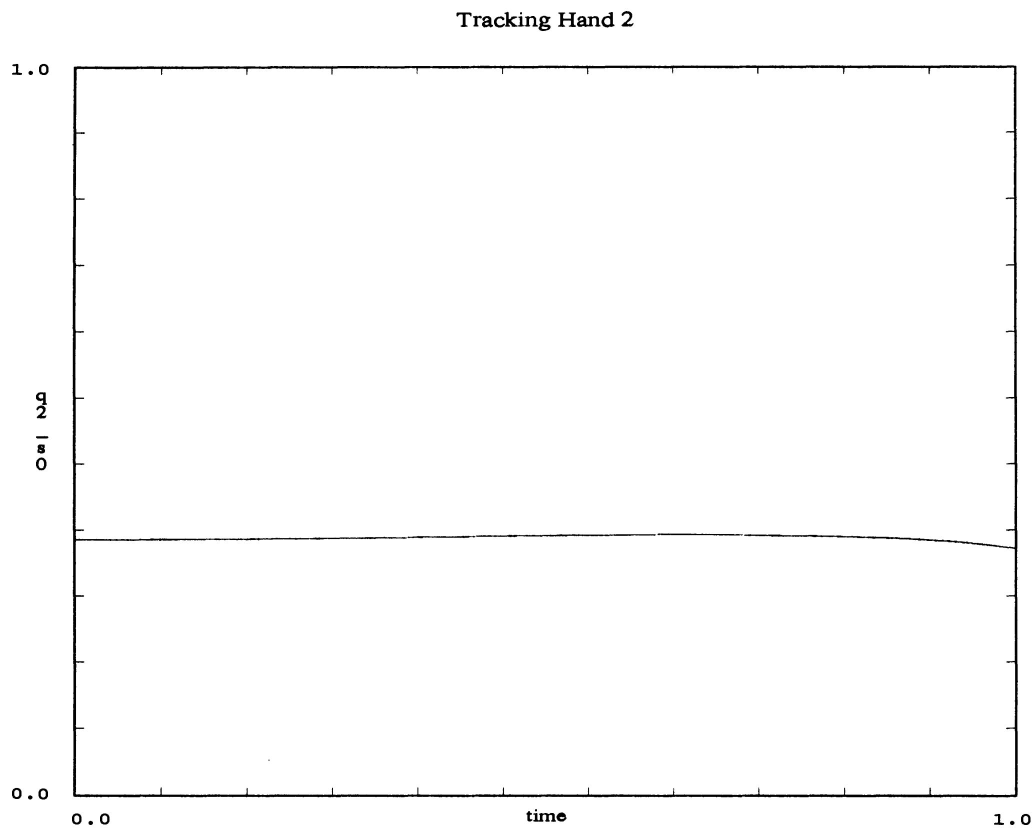

Figure 6.12: The first barycentric coordinate of $\mathbf{q_2}$

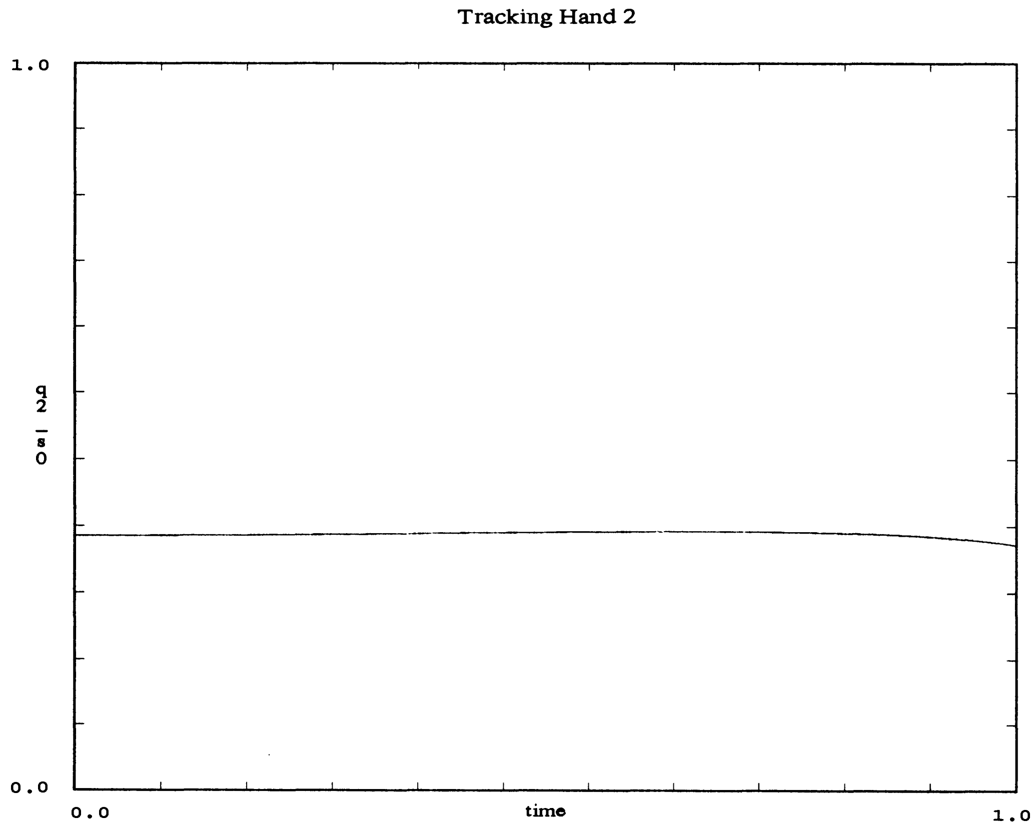

Figure 6.13: The second barycentric coordinate of $\mathbf{q_2}$

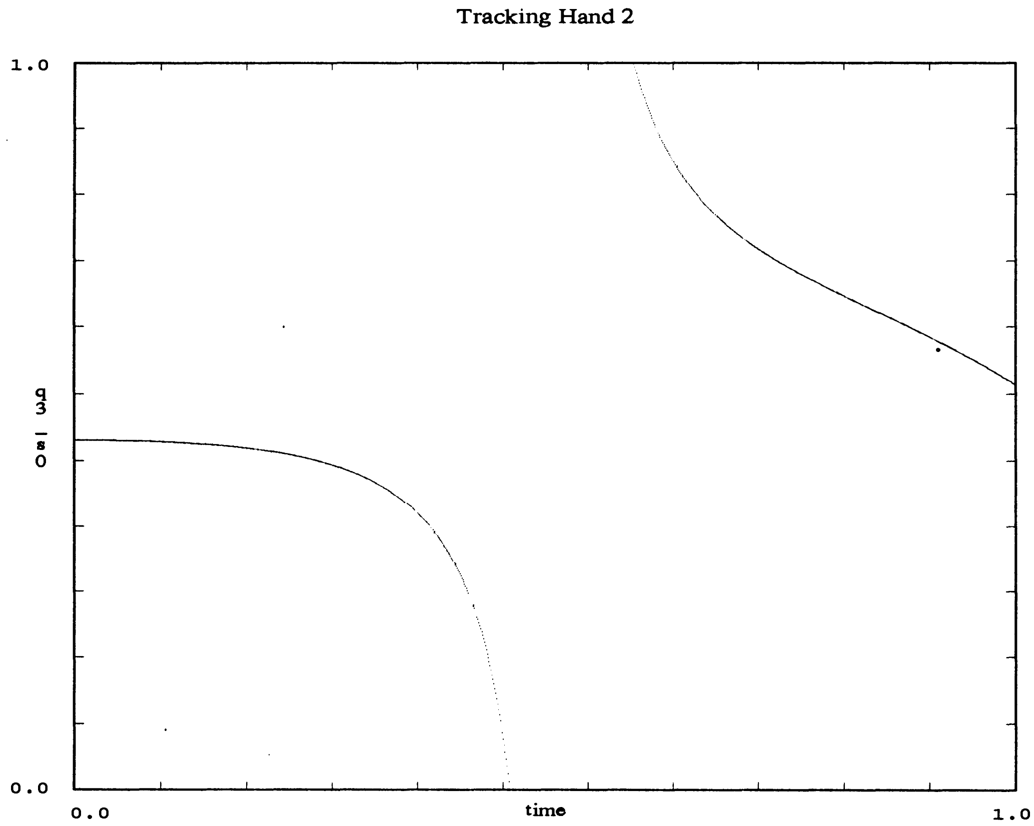

Figure 6.14: The first barycentric coordinate of $\mathbf{q_3}$

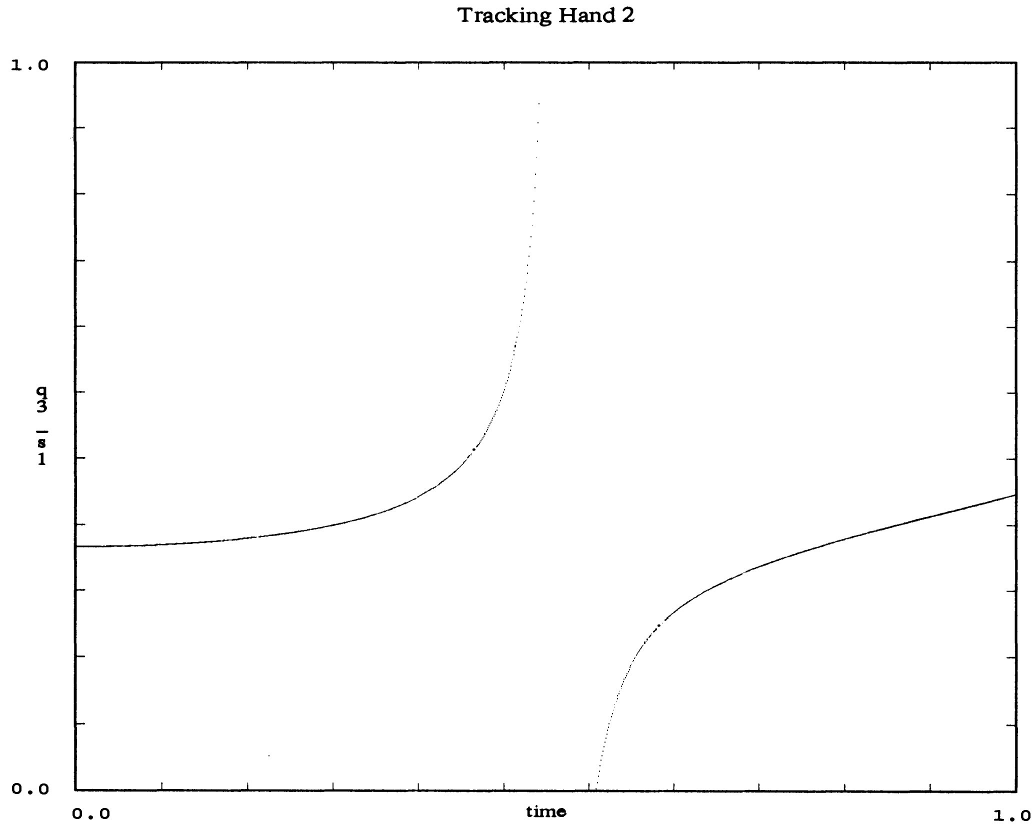

Figure 6.15: The second barycentric coordinate of $\mathbf{q_3}$

By enabling the *special events* package, the configurations for which the forces required to continue the motion exceed their allowed and the configurations for which a finger approaches an edge are detected. For the canonical example and the given initial parameters, the desired planar rotation was simulated over an interval of 10 seconds. The motion necessitated 11 switches for the tracking finger. Figure 6.17 shows a collection of frames from this simulation.

Figure 6.16: Sample frames from the simulation of the tetrahedron

6.3 Lessons

The main contribution of the simulation is the demonstration that maintaining planar rotation with the finger tracking algorithm is a feasible operation. The type of motion is important for two reasons. It is significant in itself, because many applications require planar manipulation. Planar motion is also relevant for the high-level synthesis of more complex manipulation strategies. Since any rotation can be obtained by composing planar reorientations, planar rotational motions are good primitives for task-directed manipulation algorithms. The finger tracking algorithm may also be used when the desired rotation is given as a time-varying function. Finger tracking is a practical bridge between the low-level interaction between a robot and an object and the high-level specification of tasks in terms of reorientation.

The numerical data generated by simulation is useful in detecting properties about the behavior of the system. One of the major questions about the finger tracking strategy that remains unresolved is how to choose the initial parameters for the problem in order to maximize the amount of rotation achievable by each step. Since we have no closed form solution as yet, simulation data may be used to point possible directions of further investigation. The dependency of the constrained quantities, namely the force magnitudes and the finger locations on the initial conditions can be visualized by running simulations. For example, Figure 6.18 shows several curves for the dependency of the force of the tracking finger on time. Each curve was obtained by choosing a different location for the fixed finger contact $\mathbf{q_0}$. The set of barycentryic coordinates for which the force values were plotted is $\{(0.1, 0.1), (0.2, 0.2), (0.25, 0.25), (0.3, 0.3), (0.4, 0.4), (0.5, 0.5), (0.6, 0.6)\}$. The graphs show that the time range for which the force has an admissible value depends greatly on the initial finger contacts.

The finger tracking problem is an example of how symbolic computing can be integrated with numeric computations to solve a real-world application. The symbolic solution we developed for the problem was verified by interfacing it with the simulator. The numeric information provided by the simulation was used to determine symbolic results related to the singularities and feasibility of the solution. The new results were then encoded for a continued verification. The interpretation of the simulation results guided the theoretical work and vice-versa.

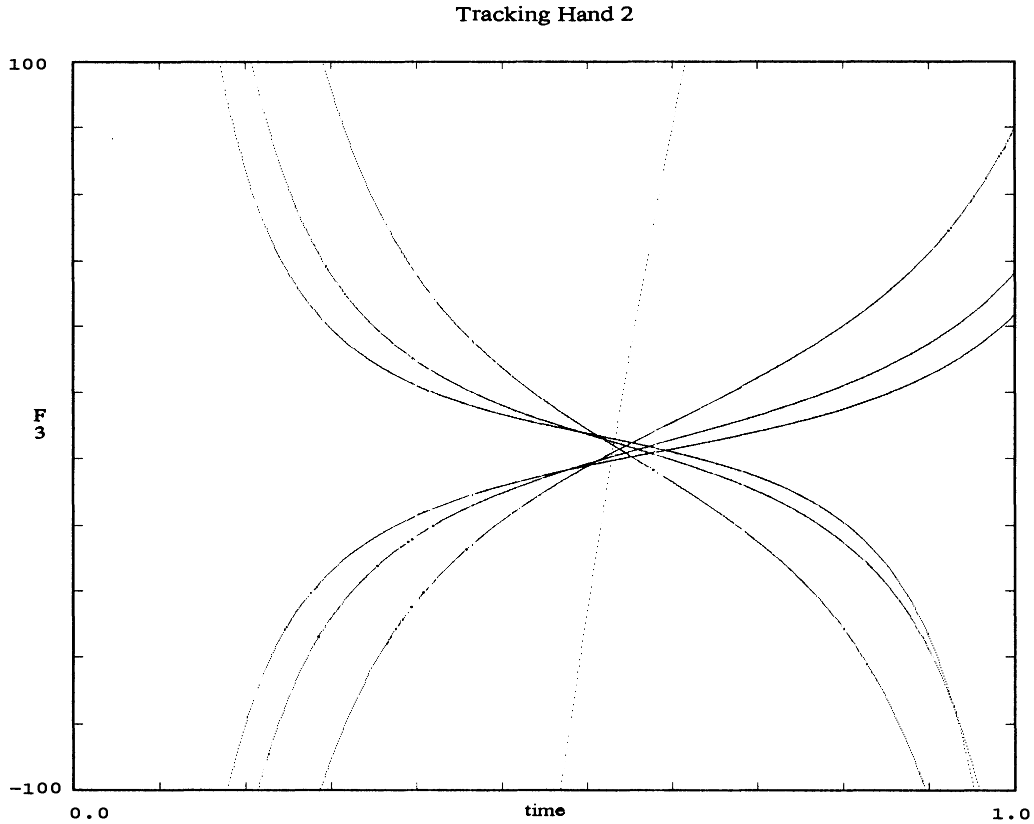

Figure 6.17: The dependency of f_3 on the location of $\mathbf{q_0}$

6.4 Summary

In this section we described a special purpose simulation system for the three dimensional finger tracking problem. The main components of the system are the module called Build System that collects and solves the equations for the constrained motion, the module called Exceptional Events, that detects when a force or a finger location exceeds its threshold and selects a new tracking finger for the continuation of the motion, the module called Convert, that converts the simulation quantities between the object-fixed and hand-fixed systems of coordinates, and Report, that produces the numerical and visualization data for the simulation. The results of the simulation for the instantiation with a regular tetrahedron being manipulated by four frictionless fingers are presented in the form of graphs and movie frames. This chapter also outlines the ways in which this implementation has contributed to the further theoretical understanding of the problem. Real achievements towards solving any real and complex problem can only be obtained by alternating theoretical and experimental investigations.

Chapter 7

Conclusions

In this thesis we have presented a framework for dexterous reorientations, and we have described a family of low-complexity algorithms.

We motivated the work by describing an algorithm for rotating polygons. The problem is to decide whether the polygon can be reoriented by any arbitrary amount and to produce a motion plan for the fingers that results in the desired reorientation of the object. Robot systems are subject to significant error in control, sensing, and geometric representation of the environment. Manipulation strategies are needed that are guaranteed to succeed in the presence of uncertainty. To this effect, an algorithmic theory of manipulation, based on high-level operations is needed. Our rotation algorithm can be viewed as a high level primitive for task-level motion planning with uncertainties. This new strategy for reorientations is derived from geometric insights. A main feature is the ability to reason about the global behavior of the object being manipulated from the local data related to the direction of the finger forces. This results in surprisingly simple finger motions and a near sensorless strategy, which enable our treatment of uncertainty.

We show that the finger motion plan based on blocking can accomplish arbitrary rotations for a class of polygons. An $O(n)$ procedure to determine whether a polygon with n edges can be rotated this way or not is provided, by building a graph with $O(n)$ edges associated with the geometry of the system and searching for a cycle. If there is no cycle, the maximal robust rotation for the given polygon can be determined. Furthermore, we calculate *stability regions* on the edges of such a polygon: regions with the property that if the grasping fingers originate anywhere within their intervals, arbitrary rotations are guaranteed.

Thus, the general procedure for the given polygon is: check if it can be rotated robustly and arbitrarily by building a small graph and searching for its cycles; if there is a cycle, determine its stability regions and generate the fingers' motion plan.

The planar reorientation algorithm has lead to an algebraic framework for describing manipulation problems. A manipulation system can be expressed algebraically in terms of the description of the class of objects being manipulated, the class of hands performing the operation, and the constraints for the motion. A manipulation plan is described as a function between these sets. The reorientation algorithm derived geometrically has been reformulated algebraically in this framework.

An outcome of this framework was the introduction of finger tracking as a high-level primitive for dexterous manipulation. To reorient an object by finger tracking with a multi-finger robot hand, a subset of the grasping fingers have a fixed position, thus constraining the motion of the object. The other grasping fingers travel of the surface of the object applying forces, thus causing the object to rotate relative to the fixed fingers. We have examined how to proceed from instantaneous analysis to large motions for finger tracking algorithms. by using Lie Algebras and computational geometry. Our work has shown that Lie Algebras provide substantial computational power for these problems. We have shown that finger tracking can be used to maintain planar rotational motions of objects whose surfaces can be specified as time-varying functions in three dimensions. This type of motion is important in itself, because many applications require planar manipulation. Moreover, planar motion is also critical for the high-level automatic synthesis of fine motion strategies. Since any rotation can be obtained by composing planar reorientations,

7.1 Future Directions

An immediate extension to this work is to develop finger tracking algorithms for different models of contact between the robot hand and the object. The three dimensional reorientation has already been extended to robot hands whose fingers have point contacts with friction. Most of this work idealizes the ways in which robots contact and sense the objects in their environment. One simplifying assumption is that the contact between the object and the hand is through point contacts. Even when friction is taken into account, this is not an accurate assumption. In practice, most contact is through surfaces rather than points. While a point contact model of a finger only models forces, most real fingers can exert forces and torques through a surface contact. We need tools for modeling soft fingers and for understanding the effects of deformation upon contact.

Another aspect of the interaction between the robot and the environment has to do with the feasibility of operations. Our current work does not address the issue of how fingers reach their desired locations. This is not a good assumption, since the fingers of mechanical hands have limited reachability. The feasibility and reachability problem must be resolved before these types of algorithms can be used in practice. The problem that should be examined is: given a limited reachability of the fingers, how does this affect the task execution and how can it be used, if necessary, to resynthesize the plan? The mathematical machinery provided by Lie Algebras was suitable for the formalization of finger tracking with unlimited finger reachability. We expect that it is a natural tool to use in order to incorporate limited finger reachability into the framework.

Attention should be given to how precisely the hand "knows" the position of objects. Determining the location of an object in the grasp is difficult. The robot needs sensory information on the object position and contact force. For some classes of objects, tactile data may by enough to determine location, but for others, such as cylinders held vertically, this is not enough. An important direction concern the extension of the stability results to the general three dimensional problem and to the other types of contacts between the fingers and the object.

Among the applications of the finger tracking algorithm, a natural problem is to determine the shape of a grasped object of unknown geometry. The finger tracking motion system could be used to determine the curvature of the grasped object at a point by doing instantaneous tracking in three different directions the recording the sensed rotation. A deeper problem is how to integrate this local process with more global strategies that could lead to methods for model acquisition.

Finger tracking is not the only manipulation mode. For certain tasks, a finger walking strategy might be preferable to a finger tracking strategy. The advantage of a finger walking procedure is that the part does not move with respect to the fingers involved in the grip, and thus it is more stable. The task specification and the geometry of the objects involved are natural candidates for deciding which method is more appropriate. Although we have defined and identified properties, we have not developed and demonstrated a mathematical formalism for finger walking. We believe that a methodology based on Lie Algebras could also be used to generate a computational model for finger walking.

Regardless of whether a finger tracking or a finger walking strategy is employed, performing large and continuous motions with robot hands is difficult. This requires the repositioning of some of the fingers and necessitates having enough fingers to stably secure the object inside the hand at all time during manipulation. The value of having more than three fingers is qualitatively obvious when observing human motions. Although we have a good understanding of how finger numbers and contact types affect grasping, we do not have a theory of how the number of fingers and contact types affect the manipulability with a certain robot hand, which is needed to fully understand the power of manipulation strategies.

For a complete validation of this work, extensive experiments should be carried with real devices. We are currently working on implementing these ideas for planar manipulation with mobile robots. During this experiment, we have discovered the need to solve a number of problems related to the real interaction between robots and objects, but our results are very promising.

Answers to the problems outlined here would take the research in robotic manipulation to a stage of automatically generating plans involving high level primitives, such as planar rotations, for complex task specifications, a clear step towards the long-standing dream of having robots capable of performing autonomously meaningful tasks.

Appendix A

Definitions

Definition A.0.1 *M is a topological manifold of dimension n if it is a Hausdorff space and every point has an open neighborhood homeomorphic to an open ball in R^n.*

A map f is a homeomorphism if both f and f^{-1} exist and are continuous. Thus, a manifold is a locally Euclidean space. When a body moves along some curve in a manifold, its velocity lies in the tangent space of the manifold. The tangent space of a manifold at a point is an n-dimensional space representing the linear approximation to the manifold near the point and it may be viewed as a set of equivalence classes. More formally, let M be a manifold, p a point in M, and γ_1, γ_2 curves in M. We say that γ_1, γ_2 belong to the same equivalence class $[\gamma]_p$ if for any differentiable function $f : U \to R, U$ a neighborhood of $p \in M$, we have $\frac{d}{dt}|_{t=0} f \circ \gamma_1 = \frac{d}{dt}|_{t=0} f \circ \gamma_2$. Then

Definition A.0.2 *The tangent space of a manifold M at point p is $T_pM = \{[\gamma]_p\}$ where γ is a curve in M with $\gamma(0) = p$.*

Definition A.0.3 *A Lie group G is a group which is also a manifold, and for which the composition and the inversion operations are smooth.*

Example A.0.1 *The circle $S^1 = \{(x, y) \in R^2 | x^2 + y^2 = 1\}$.*

Example A.0.2 *R^n and $SO(n)$.*

Definition A.0.4 *A left translation of a group G is a map $L_g : G \to G$ with $g \in G$, so that for every element h in G, $L_g(h) = gh$.*

Definition A.0.5 *A left invariant vector field is a vector field X satisfying $X_{gh} = L_{g*}(X_h)$, where $L_{g*} : T_hG \to T_{gh}G$ is the map induced by L_g between the corresponding tangent spaces.*

Definition A.0.6 *The Lie algebra of a Lie group G is the set of all left invariant vector fields on G.*

Example A.0.3 *The Lie algebra of $SO(n)$ is the set of $n \times n$ skew symmetric matrices.*

Proof: Let $\gamma(t)$ be a curve in $SO(n)$, with $\gamma(0) = \mathbf{I}$, the identity. From the definition of $SO(n)$ we have that $[\gamma(t)]^t\gamma(t) = \mathbf{I}$, so differentiating for $t = 0$ it follows that $[\gamma'(0)]^t\gamma(0) + [\gamma(0)]^t\gamma'(0) = 0$. In other words, $[\gamma'(0)]^t + \gamma'(0) = 0$, *i.e.* $\gamma(0)$ is skew symmetric. □

Definition 2.2.8 shows directly the connection a Lie group and its Lie algebra. An equivalent definition for a Lie algebra can be given in terms of axioms. A Lie algebra is a vector space with a bilinear operation which is antisymmetric and which satisfies the Jacobi identity. If the additional operation id a map $V \times V \to V$ taking the pair $(\mathbf{v_1}, \mathbf{v_2})$ to $[\mathbf{v_1}, \mathbf{v_2}]$, then the axioms are:

1. Bilinearity: $[\alpha_1\mathbf{v_1} + \alpha_2\mathbf{v_2}, \mathbf{v_3}] = \alpha_1[\mathbf{v_1}, \mathbf{v_3}] + \alpha_2[\mathbf{v_2}, \mathbf{v_3}]$, and $[\mathbf{v_1}, \alpha_2\mathbf{v_2} + \alpha_3\mathbf{v_3}] = \alpha_2[\mathbf{v_1}, \mathbf{v_2}] + \alpha_3[\mathbf{v_1}, \mathbf{v_3}]$.

2. Antisymmetry: $[\mathbf{v_1}, \mathbf{v_2}] = -[\mathbf{v_2}, \mathbf{v_1}]$.

3. Jacobi identity: $[\mathbf{v_1}, [\mathbf{v_2}, \mathbf{v_3}]] + [\mathbf{v_2}, [\mathbf{v_3}, \mathbf{v_1}]] + [\mathbf{v_3}, [\mathbf{v_1}, \mathbf{v_2}]] = \mathbf{0}$.

An example of an operation that satisfies the Jacobi identity is the vector crossproduct.

Every finite dimensional Lie algebra is the Lie algebra of some Lie group. The Lie algebra of a Lie group and the tangent space at the identity of the group are isomorphic. Moreover, because of left translations, all the tangent spaces of the Lie group may be identified with the Lie algebra.

Bibliography

[AB89] F. Avnaim and J. D. Boissonnat. Polygon placement under translation and rotation. *Informatique Thóretique et Application*, pages 5–28, 1989.

[AM77] L. Auslander and R. MacKenzie. *Introduction to Differentiable Manifolds.* Dover Publications, Inc., New York, 1977.

[Bas88] M. Bastuscheck. On the stability of grasping: three fingers and a planar polygon. In *Proceedings of the IEEE Conference of Robotics and Automation*, 1988.

[BFG85] P. Baker, S. Fortune, and E. Grosse. Stable prehension with three fingers. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 114–120, 1985.

[BGM+92] A. Back, J. Guckenheimer, M. Meyers, F. Wicklin, and P. Worfolk. DSTOOL: Computer assisted exploration of dynamical systems. *Notices of the AMS*, pages 303–309, 1992.

[BLP85] R. Brooks and T. Lozano-Peréz. A subdivision algorithm in configuration space for findpath with rotations. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 224–233, 1985.

[BR] A. Back and D. Rus. Singularities in constraint-based manipulation. In *(in preparation)*.

[Bri89] A. Briggs. An efficient algorithm for one-step planar compliant motion with uncertainty. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 187–196, 1989.

[Bro87] D. Brock. *Enhancing the dexterity of a robot using controlled slip.* Technical Report MIT AI-TR-992, 1987.

[Bro88] R. Brost. Automatic grasp planning in the presence of uncertainty. *International Journal of Robotics Research*, pages 3–17, 1988.

[Bro91] R. Brost. *Analysis and Planning of Planar Manipulation Tasks.* Ph.D. dissertation, Carnegie Mellon University, 1991.

[Buc89] S. Buckley. Planning compliant motion strategies. *International Journal of Robotics Research*, 8(5):28–44, 1989.

[Can88] J. Canny. *The Complexity of Robot Motion Planning.* The MIT Press, Cambridge, Massachusetts, 1988.

[CD88] J. Canny and B. Donald. Simplified Vornoi diagrams. *Discrete and Computational Geometry*, 3:219–236, 1988.

[CR87] J. Canny and J. Reif. New lower bound techniques for robot motion planning. In *Proceedings of the 28^{th} Annual Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[Cre89] J. Cremer. *An Architecture for General Purpose Physical System Simulation - Integrating Geometry, Dynamics, and Control.* Ph.D. dissertation, Cornell University, 1989.

[CS89] J. Cremer and J. Stewart. Using the newton simulator as a testbed for control. In *Proceedings of the IEEE Conference of Robotics and Automation*, 1989.

[Cut85] M. Cutkoski. *Grasping and Fine Manipulation for Automated Manufacturing.* Ph.D. dissertation, Carnegie Mellon University, 1985.

[Don87a] B. Donald. *Error Detection and Recovery in Robotics.* Springer-Verlag, New York, 1987.

[Don87b] B. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, pages 295–353, 1987.

[EM88] M. Erdmann and M. Mason. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, pages 369–379, 1988.

[Erd86] M. Erdmann. Using back projections for fine motion planning with uncertainty. *International Journal of Robotics Research*, pages 19–45, 1986.

[Erd89] M. Erdmann. *On Probabilistic Robot Strategies.* Ph.D. dissertation, M.I.T., 1989.

[Fea86] R. Fearing. Simplified grasping and manipulation with dexterous robot hands. *IEEE Journal of Robotics and Automation*, pages 188–195, 1986.

[Fea87] R. Fearing. Some experiments with tactile sensing during grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1637–1643, 1987.

[FHS88] J. Friedman, J. Hershberger, and J. Snoeyink. Input-sensitive compliant motion in the plane. In *Second Scandinavian Workshop on Algorithmic Theory*, pages 445–459, 1988.

[FW88] S. Fortune and G. Wilfong. Planning constrained motion. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 445–459, 1988.

[Gil74] R. Gilmore. *Lie groups, Lie algebras, and some of their applications.* John Wiley, 1974.

[GM90] K. Goldberg and M. Mason. Bayesian grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1264–1269, 1990.

[Gol90] K. Goldberg. *Stochastic Plans for Robot Manipulation.* Ph.D. dissertation, Carnegie Mellon University, 1990.

[Goy89] S. Goyal. *Planar Sliding of a Rigid Body with Dry Friction: Limit Surfaces and Dynamics of Motion.* Ph.D. dissertation, Cornell University, 1989.

[HH86] C. Hoffmann and J. Hopcroft. Simulation of physical systems from geometric models. *IEEE Journal of Robotics and Automation*, pages 194–206, 1986.

[HKK91] J. Hopcroft, J. Kearney, and D. Krafft. A case study of flexible object manipulation. *International Journal of Robotics Research*, pages 41–50, 1991.

[HLMT90] J. Hong, G. Lafferriere, B. Mishra, and X. Tan. Fina manipulation with multifinger hands. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990.

[HSS87] J. Hopcroft, J. Schwartz, and M. Sharir, editors. *Planning, Geometry, and Complexity.* Ablex Publishing Corporation, Norwood, New Jersey, 1987.

[HW86a] J. Hopcroft and G. Wilfong. Motion of objects in contact. *International Journal of Robotics Research*, 4(4):32–46, 1986.

[HW86b] J. Hopcroft and G. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal of Computing*, 15(3):768–785, 1986.

[JIK+86] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers. The design of the Utah/MIT hand. In *Proceedings of the IEEE Conference of Robotics and Automation*, 1986.

[JL87] J. Jameson and L. Leifer. Automatic grasping: an optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 806–814, 1987.

[KMY92] D. Kirckpatrick, B. Mishra, and C. Yap. Quantitative Steinitz's theorems with applications to multifinger grasping. *Discrete and Computational Geometry*, pages 295–318, 1992.

[Lat89] J.C. Latombe. *Motion planning with uncertainty: On the preimage backchaining approach.* MIT press, Cambridge, Massachusetts, 1989.

[Lat91] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[LCS89] Z. Li, J. Canny, and S. Sastry. On motion planning for dexterous manipulation, part i: the problem formulation. In *Proceedings of the IEEE Conference of Robotics and Automation*, pages 775–780, 1989.

[Li89] Z. Li. *Kinematics, PLanning, and Control of Dexterous Robot Hands.* Ph.D. dissertation, Berkeley University, 1989.

[LP87] T. Lozano-Peréz. A simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation*, pages 224–238, 1987.

[LPMT84] T. Lozano-Peréz, M. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, pages 3–24, 1984.

[LS88] Z. Li and S. Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal of Robotics and Automation*, pages 32–43, 1988.

[LS89] P. Li, Z. Hsu and S. Sastry. Grasping and coordinated manipulation by a multifinger robot hand. *International Journal of Robotics Research*, pages 33–50, 1989.

[Mas81] M. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:418–432, 1981.

[Mas86] M. Mason. Mechanics and planning of manipulator and pushing operations. *International Journal of Robotics Research*, pages 53–71, 1986.

[Mon88] D. Montana. The kinematics of contact and grasp. *International Journal of Robotics Research*, pages 17–32, 1988.

[MP89] X. Markenscoff and C. Papadimitriou. Optimum grip of a polygon. *International Journal of Robotics Research*, pages 17–29, 1989.

[MS85] M. Mason and K. Salisbury. *Robot hands and the mechanics of manipulation.* The MIT Press, Cambridge, Massachusetts, 1985.

[MSS87] B. Mishra, J. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, pages 541–558, 1987.

[Nat88] B. Natarajan. The complexity of fine-motion planning. *International Journal of Robotics Research*, pages 36–41, 1988.

[Nat89] B. Natarajan. Some paradigms for the automated design of parts feeders. *International Journal of Robotics Research*, pages 98–109, 1989.

[Ngu88] V. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, pages 3–16, 1988.

[Ngu89] V. Nguyen. Constructing stabel grasps. *International Journal of Robotics Research*, 8(1):26-37, 1989.

[PS88] M. Peshin and A. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, pages 524–531, 1988.

[PS90] Y. Park and G. Starr. Grasp synthesis for polygonal objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1574–1580, 1990.

[PT89] J. Pertin-Troccaz. *Grasping: a state of the art, In The Robotics Review I, edited by O. Khatib, J. Craig, and T. Lozano-Peréz.* MIT Press, Cambridge, Massachussetts, 1989.

[RC81] M. Raibert and J. Craig. Hybrid force/position control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, pages 126–133, 1981.

[Rei87] J. Reif. Complexity of the generalized mover's problem. In *Planning, Geometry, and Complexity of Robot Motion, edited by J. Hopcroft, J. Schwartz, and M. Sharir*, pages 267–281, 1987.

[Rus90] D. Rus. Dexterous rotations of polygons. In *Proceedings of the Canadian Computatinal Geometry Conference*, pages 157–162, 1990.

[Rus91] D. Rus. A framework for dexterous manipulation using lie algebras. In *Proceedings of the Second Conference on Algebraic Methodology and Software Technology*, 1991.

[Ste65] S. Sternberg. *Lectures on differential geometry.* Prentice Hall, 1965.

[SY87] J. Schwartz and C. Yap, editors. *Algorithmic and Geometric Aspects of Robotics.* Lawrence Erlbaum Associates, Hillside, New Jersey, 1987.

[TAP88] J. Trinkle, J. Abel, and R. Paul. An investigation of frictionaless enveloping grasping in the plane. *International Journal of Robotics Research*, pages 33–51, 1988.

[TLPM87] P. Tournassoud, T. Lozano-Peréz, and E. Mazer. Regrasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1924–1928, 1987.

[TP90] J. Trinkle and R. Paul. Planning for dexterous manipulation with sliding contacts. *International Journal of Robotics Research*, 1990.

[Wit77] J. Wittenburg. *Dynamics of Systems of Rigid Bodies.* B. G. Teubner, Stuttgart, 1977.

[Xav92] P. Xavier. *Provably-Good Approximation Algorithms for Optimal Kinodynamic Robot Motion Planning.* Ph.D. dissertation, Cornell University, 1992.

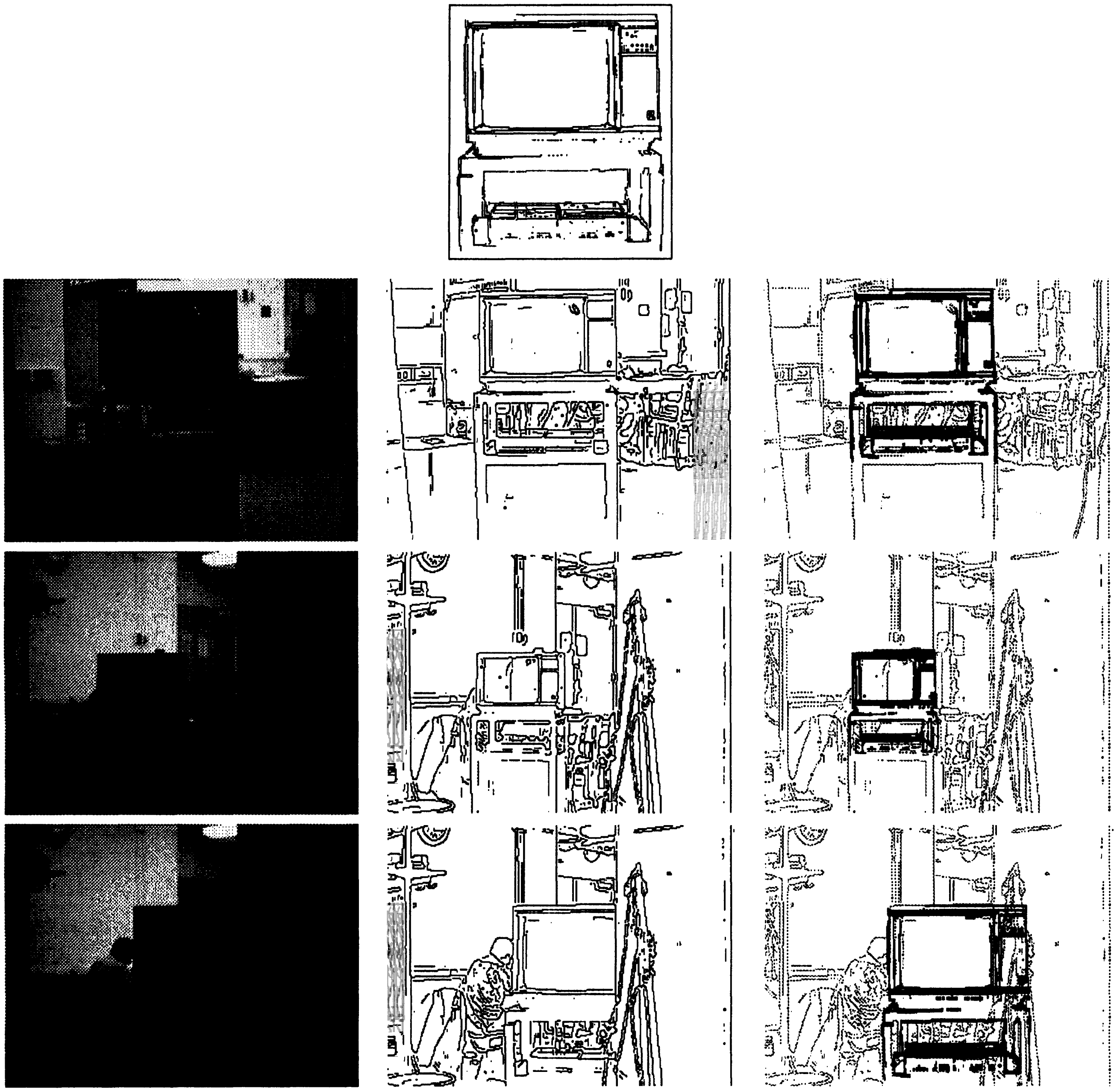

Figure 1: The model at full scale, and the three test images for the first example.

the image. If it rotates slightly either about a vertical axis, or about a horizontal axis which is perpendicular to the camera axis, this will appear as a scaling in x or y, respectively. These two effects can be produced by the camera base panning or tilting, as it might if mounted on a pan-tilt head.

We now consider some examples in order to illustrate the performance of the Hausdorff distance methods developed above. The first test model is shown in the top row of Figure 1.